Syllabus

Sunday, January 9, 2022 3:37 PM

25% — Midterm 1 ("Quiz")
25% — Midterm 2 ("Quiz")
25% — Final ("Quiz")

→ One sheet of paper allowed (A4, 8.5"×11"). Both sides. Handwritten. (2 sheets total)
→ Two sheets of paper allowed (A4, 8.5"×11"). Both sides. Handwritten. (4 sheets total)

→ Allowed blank sheets of papers for extra scratch work. Must show front/back for proctoring.

Miss a quiz, too bad. But in some situations, prof may input a score for missed quiz, based on previous scores.

15% — Homeworks { Every week. Peer graded. 2 lowest are dropped.

8% — Course Project { Peer graded.

2% — Syllabus Quiz

Must finish:
- Onboarding Quiz (ungraded) — Due Jan 24, 2AM
- Syllabus Quiz (2%, unlimited tries) — Due Jan 24, 2AM
- Piazza Registration (part of onboarding)

Table of Contents

Friday, March 4, 2022 3:44 PM

Note

characteristic/measurement of data → column of data
= attribute
= feature
= predictor
= covariate

Week 1: Classification, Scaling

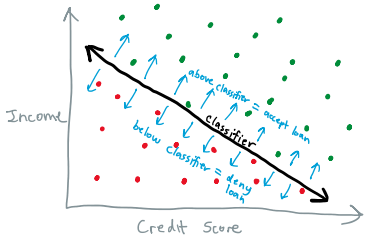
Tuesday, January 11, 2022 8:50 PM

Classification: the separation of data into two or more categories, or (a point's classification) the category a data point is put into

Classifier: A boundary that separates the data in two or more categories. Also (more generally) an algorithm that performs classification.

Support Vector Machines (SVM): classification algorithm that uses a boundary to separate the data into two or more categories ("classes")

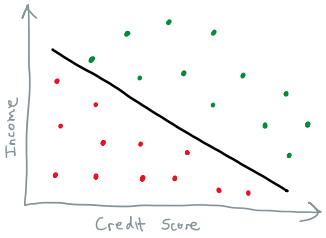
→ basically algorithm (math) to draw lines between points.



Note:
- classifier doesn't need to be in a straight line → "non-linear classifier"
- can have multiple dimensions/attributes (not just red/green two, in example)
- some methods give probabilistic answers, i.e. "accept" / "deny" vs. "37% chance to deny"

If classifier isn't perfect (i.e. mix of red & green), can create soft classifier.

SVM Mathematics



n = number of data points
m = number of attributes

Eye Color	Hair Color
Brown	Black
Black	Black
Blue	Blonde

⇒ $n = 3$ (colors), $m = 2$ (Eye color, hair color)

x_{ij} = ith attribute of jth data point

x_{1j} = credit score of person j

x_{2j} = income of person j

	Credit Score (x_{1j})	Income (x_{2j})
point 1	600 = x_{11}	50,000 = x_{21}
point 2	650 = x_{12}	55,000 = x_{22}
point 3	700 = x_{13}	60,000 = x_{23}

y_j = response for data point j

i.e. $y_j = \begin{cases} 1, & \text{if data point } j \text{ is green} \\ -1, & \text{if data point } j \text{ is red} \end{cases}$

arbitrarily set as ±1; can be any ± constant

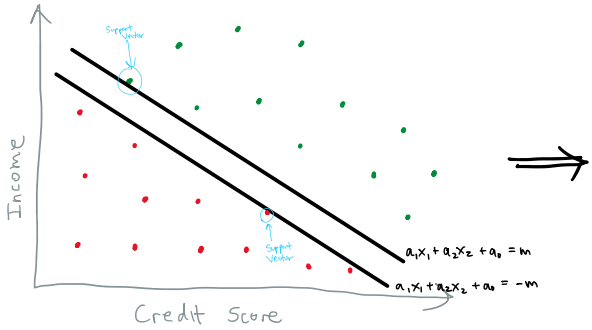
Line: $a_1x_1 + a_2x_2 + \dots + a_mx_m + a_0 = 0$ (from generic distance of a line equation)

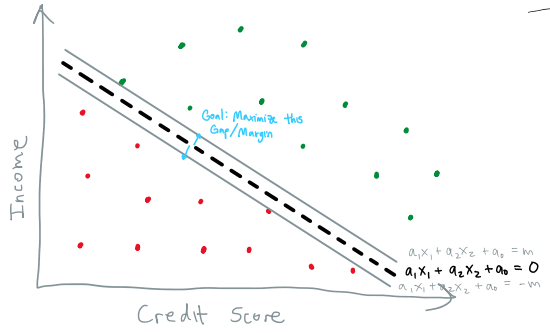
In other words: $\sum_{i=j}^{m} a_i x_i + a_0 = 0$

- a_i = coefficient defining classifier (like a "summary" of n data points (row) for every m attribute (column) vs. one-by-one)
- a_0 = intercept of classifier

Note: near-zero a_i coefficients are likely not relevant for classification.

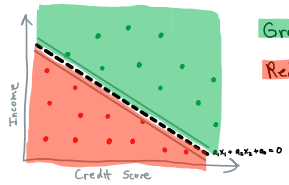
Can draw 2 parallel lines so a_0 is the classifier in the middle.





The goal is to find two parallel lines that keep red points on one side and green points on the other; the classifier would be right in the middle of these two lines.

More Equations.....



Green Points: $a_1x_{1j} + a_2x_{2j} + \dots + a_mx_{mj} + a_0 \geq 1$

Red Points: $a_1x_{2j} + a_2x_{2j} + \dots + a_mx_{mj} + a_0 \leq -1$

could be any m number... picked 1 for scaling

ALL POINTS: $(a_1x_{1j} + a_2x_{2j} + \dots + a_mx_{mj} + a_0)\,y_j \geq 1$

(from linear algebra) Distance between two solid lines $= \frac{2}{\sqrt{\sum_i (a_i)^2}}$ ..... Remember the goal is to maximize the margin. ⇒ ↑ distance = (constant) / ↓ denominator ⇒ So, minimize a! ⇒ $\min \sum_i (a_i)^2$

⇒ Minimize $_{a_0, \dots, a_m} \sum_{i=1}^{m} (a_i)^2$, subject to $(a_1x_{1j} + a_2x_{2j} + \dots + a_mx_{mj} + a_0)\,y_j \geq 1$ for each data point.

So, need to find values a_0 thru a_m that will maximize the margin, but can only choose values of coefficient a that correctly separate all the points.... Software does this!

Summary:

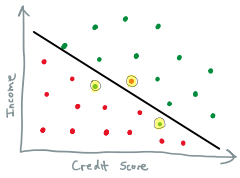
Given data $\{x_{ij}\}$ where x_{ij} is the jth factor of data point i:

- Classifier is the line: $0 = a_0 + \sum_j a_j x_j$
- Maximize gap between lines by minimizing $\sum_j a_j^2$

Soft Classifier

Use if not possible to perfectly separate red and green points.

Need to account for errors in classification, and trade off minimizing the errors we make vs. maximizing the margin.



Errors

If the point is on the Correct side of the line: $\sum_{i=1}^{m}(a_i x_{ij} + a_0)y_j \geq 1$ // $\sum_{i=1}^{m}(a_i x_{ij} + a_0)y_j - 1 \geq 0$ = true

(remember: anything)

If the point is on the Wrong side of the line: $\sum_{i=1}^{m}(a_i x_{ij} + a_0)y_j < 1$ // $\sum_{i=1}^{m}(a_i x_{ij} + a_0)y_j - 1 < 0$

Error Magnitude... ↑Error = ↑ Margin

General Error for data point j: $\max\left\{0,\ 1 - \left(\sum_{i=1}^{m} a_i x_{ij} + a_0\right)y_j\right\}$

The error is whichever one is largest, zero, or that summation

Trade-Off: ↑margins vs. ↓Errors

Sometimes, having some error will increase our margin.
Other times, too many mistakes will not result in an inconclusive margin.

How do we calculate this trade-off?

This is called the **Support Vector Machine (SVM) Model**:

$$\underset{a_0,\cdots,a_m}{\text{minimize}} \sum_{j=1}^{n} \max\left\{0,\ 1 - \left(\sum_{i=1}^{m} a_i x_{ij} + a_0\right)y_j\right\} + \lambda \sum_{i=1}^{m} (a_i)^2$$

$\underset{a_0,\cdots,a_m}{\text{minimize}}$ $\sum_{j=1}^{n} \max\left\{0,\ 1 - \left(\sum_{i=1}^{m} a_i x_{ij} + a_0\right)y_j\right\}$ + λ $\sum_{i=1}^{m} (a_i)^2$

Error — Margin

Pick a value:

↑λ = ↑Margin (towards ∞) → ↑error → a large margin outweighs avoiding mistakes and classifying known data points.

↓λ = ↓Margin (towards 0) → ↓error → minimizing mistakes and classifying known data points outweighs a large margin

Intercept a_0: "risk" tolerance, i.e. ↑a_0 if classifying eating wild mushroom vs not eating it.

Range: $\begin{cases} a_0 + m \ / \ a_0 - m \\ \text{if } a_0 + 1 \ / \ a_0 - 1 \end{cases}$

ex) If giving a bad loan is twice as costly as withholding a good loan:

$\Rightarrow a_1x_1 + a_2x_2 + \ldots + a_mx_m + \left[\frac{2}{3}(a_0 - 1) + \frac{1}{3}(a_0 + 1)\right] = 0$ ⟿ 1/3 vs 2/3 ... arbitrary

$[a_0 - \frac{1}{3}]$

(in soft classification)
Adding a multiplier for each type of error, with a larger penalty the less we want to accept mis-classifying that type of point:

$$\underset{a_0,\cdots,a_m}{\text{minimize}} \sum_{j=1}^{n} m_j \max\left\{0,\ 1 - \left(\sum_{i=1}^{m} a_i x_{ij} + a_0\right)y_j\right\} + \lambda \sum_{i=1}^{m} (a_i)^2$$

$\begin{cases} m_j > 1 \text{ for more costly errors} \\ m_j < 1 \text{ for less costly errors} \end{cases}$

Scaling Data:

We want to ↑margin by $\underset{a_0,\cdots,a_m}{\text{minimize}} \sum_{i=1}^{m} (a_i)^2$, but what if a_m values are magnitudes apart for each attribute?

Example: $\begin{cases} \text{Income in } \$10{,}000 - \$1{,}000{,}000 \text{ range} \\ \text{Credit score in } 150 - 850 \text{ range} \end{cases}$ ⇒ income a_1 (income) will dwarf a_2 (credit score)! ⇒ solution: **Scale data**, so that the orders of magnitude are approx the same.

{ Some softwares automatically do this; others don't, so do it yourself.

ex) Suppose we try coefficients 5 for income and 701 for credit score, with intercept of a million.

$\Rightarrow 0 = a_0 + \sum_j a_j x_j \Rightarrow 0 = a_0 + a_1 x_1^{\text{income}} + a_2 x_2^{\text{credit score}}$

$\Rightarrow 0 = 1{,}000{,}000 + 5x_{\text{income}} + 701\,x_{\text{credit score}}$ is the SVM.

Sum of squared coefficients:

$\Rightarrow \sum_j a_j^2 \Rightarrow 5^2 + 701^2 \Rightarrow 491{,}025$

Let's say credit score coefficient increased by one...

↳ $5^2 + 701^2 = 491{,}426$ → +1401 increase!

Big increase for a small change in coefficient.

... more than one! And a different line!

$\Rightarrow \sum_j a_j^2 \Rightarrow 5^2 + 701^2 \Rightarrow 491{,}025$

Let's say credit score coefficient increased by one....

$\hookrightarrow 5^2 + 701^2 = 491{,}426 \rightarrow +1401$ increase!

Big increase for a small change in coefficient.

To match this number, x_{income} would have to be 37.8 ... much more than one! And a different line!
(600% change)

Conclusion: SDM model won't work well without scaling.

Adjusting the Data (Scaling)

(Common scaling: data between 0 and 1 ...) → which method to use?

Scaling by factor:

Let:
- $x_{min_j} \Rightarrow \min_i x_{ij}$ (smallest # in data set)
- $x_{max_j} \rightarrow \max_i x_{ij}$ (biggest # in data set)

So, for each data point j:

$$x_{ij}^{scaled} = \frac{x_{ij} - x_{min_j}}{x_{max_j} - x_{min_j}} \xrightarrow{\text{in other words}} x_{scaled} = \frac{x - x_{smallest}}{x_{biggest} - x_{smallest}}$$

(General) Scaling between b and a:

$$\Rightarrow x_{ij}^{scaled[b,a]} = x_{ij}^{scaled[0,1]}(a-b) + b$$

Scaling to a normal distribution ("Standardization")

Common scaling: mean (μ) = 0, standard deviation (σ) = 1

Let:
- Factor j have a mean of $\mu_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}$
- Factor j has a standard deviation σ_j

So, for each data point j:

$$x_{ij}^{standardized} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

IN GENERAL....

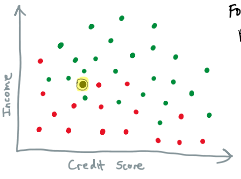
Use scaling for bounded ranges, (i.e. RGB (0-255); SAT scores (200-800))
& standardization for others. (i.e. principal component analysis; clustering)

But sometimes not clear. Try both!

Always scale/standardize your data!

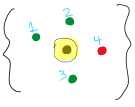
k-Nearest Neighbor (kNN) Model

Assume a new data point comes in, and there is no red/green classification on it yet.
Count k number of dots around it, and classify it the same as the greatest dot classification.



For example, look at the unclassified black dot.
Perform the kNN with K = 4...

Here are the 4 nearest dots



There are more green dots than red dots. Therefore, this data point is considered green.

→ "k closest points" number

Note: can have multiple colors, and k can be any number.

What if it's not clear which dots are closest? → Calculate it!

- Straight-line distance $= \sqrt{\left(\sum_{i=1}^{n} |x_i - y_i|^2\right)}$
- Weighing attributes by importance: $\sqrt{\left(\sum_{i=1}^{n} w_i |x_i - y_i|^2\right)}$
  (w_i = weight, set yourself; w_i = 0 for unimportant attributes)

What's a good value of k? ... We'll learn about that! → "Validation" aka measuring the quality of a model

Classification Topic Summary

- Divide data points into groups based on similarity (comes up a lot in analytics)
- Graphical intuition (look at a graph and know classifiers)
- Mathematic Solution Methods
  - Support Vector Machine (SVM)
  - k-nearest-neighbor (KNN)

  → These are ML algorithms!

Additional subjects ...
- Data terminology & data types
- Validation
- Distance metrics
- Confusion matrices

Week 2: Validation, Clustering

Tuesday, January 25, 2022 6:06 PM

Validation — checking how good a model is

- "How good is the model?"
- "How accurate is it?"
- "How well does it predict X?"

Validation does NOT depend on data point accuracy for classifier....

ex) If 24 data points and 3 are incorrectly classified, it is ✗ INCORRECT to assume 21/24 = 87.5% accuracy... too optimistic!

Data has two types of patterns:

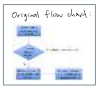
1) Real effect: real relationship between attributes and response
2) Random effect: random, but looks like a real effect

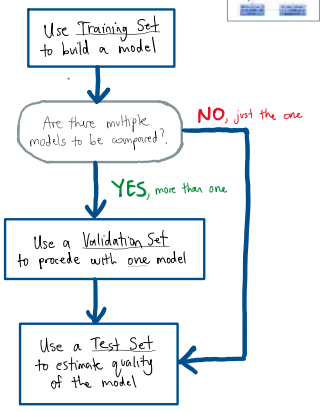
} Problem: hard to know which one it is...

Solution: fit matches on both real & random effects
→ Real effects: same in all data sets
→ Random effects: different in all data sets

Training, Validation, & Test Data Set

My flow chart:

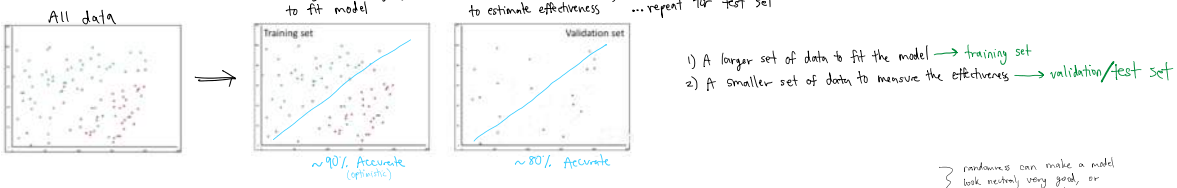




Need to split data for training, validating, and testing.
Using all data for training is a terrible approach because then there wouldn't be any data left to test the method.
Likewise, reusing same data for training & testing is a bad idea because we need to know how the method will work on data it wasn't trained on.

- **Training Set**: building models
- **Validation Set**: picking one model (if applicable)
- **Test Set**: estimating performance of model

For example.... (let's say multiple models)



1) A larger set of data to fit the model → training set
2) A smaller set of data to measure the effectiveness → validation/test set

Since observed performance = real quality + random effects, a high performing model is more likely to have above-average random effects

} randomness can make a model look mediocre, very good, or very bad. Is random!

So, observed performance of a given model is probably too optimistic.

Splitting: how to split data into training, validation, and test sets

How much data goes in each set? Generally / rule of thumb....

- 1 model: 70-90% for training, 10-30% for test → let it equal 100%
- 2+ models: 50-70% for training, and the rest is split equally between validation and test → i.e. 60% training, 20% validation, 20% test

Knowing the total number of data points, how are they split into each set?

ex) Let's say 1000 data points: 60% training, 20% validation, 20% test

Method 1: Random

- Randomly choose 600 data points for training
- Randomly choose 200 (of the remaining 400) data points for validation
- The remaining 200 data points make up the test set

Pros: eliminates bias

Cons: due to randomness, data can be unevenly separated
i.e. if 10 yrs of data, may have data sets with much more data from specific years

Method 2: Rotation

- Take turns selecting points

i.e. 5 data point rotation sequence could look like:
(1) Training → (2) Validation → (3) Training → (4) Test → (5) Training → repeat....

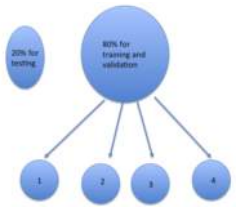
Pros: can ensure data is (equally separated) partitioned evenly
i.e. if 10 yrs of data, can ensure data sets include all years

Cons: can introduce bias
i.e. if 5 pt rotation, and based on Mon-Fri, all Mondays would be in one set, Tuesdays in another, etc.

Training → Validation → Training ...

Cross Validation: use if important data only appears in the validation or test sets (not training set)

Denoted as "k-fold cross validation"

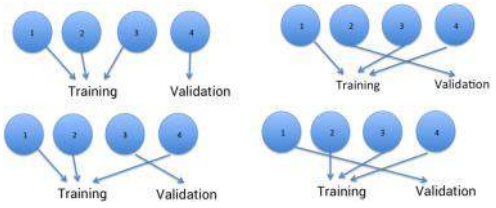


Let's say k = 4 (four-fold cross validation)
Can divide the training/validation set into 4, running the model (flowchart) with each data set k (4) times.

→ only one to be validated at a time

No standard number to use as k, but k=10 is common.

Visual Example:



Basically, for each of the k parts:

- Train the model on all the other parts
- Evaluate (validate) it on the remaining part

Then, average the k evaluations to estimate the models quantity.
Can compare different model types and use this value to pick the best model method!

→ But for SVM, do NOT average. Instead, train the model again using all the data.

Clustering → taking a set of data points and dividing them up into groups; each group contains points that are close to each other or similar.

Examples of use:
- targeted marketing
- personalized medicine
- locating facilities
- image analysis
- Data investigation

Distance Norms

(2-norm) Euclidean (straight-line) distance:

$$\text{distance} = \sqrt{(x_1-y_1)^2 + (x_2-y_2)^2}$$

$$\text{distance} = |x_1-y_1| + |x_2-y_2|$$

$$= \sum_{i=1}^{n} |x_i-y_i|$$

let $2 \to r$

(p-norm) Minkowski distance:

$$\text{distance} = \sqrt[p]{|x_1-y_1|^p + |x_2-y_2|^p}$$

$$\text{distance} = \sqrt[p]{|x_1-y_1|^p + |x_2-y_2|^p}$$

$$= \sqrt[p]{\sum_{i=1}^{n} |x_i-y_i|^p}$$

let $p \to \infty$

∞-norm distance

$$\text{distance} = \sqrt[\infty]{\sum_{i=1}^{n} |x_i-y_i|^\infty}$$

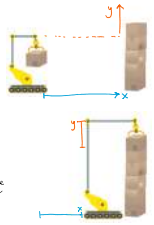
$$|x_1-y_1|^\infty + |x_2-y_2|^\infty + \dots + |x_n-y_n|^\infty$$ sum equals the largest $|x_i-y_i|^\infty$!

$$\sqrt[\infty]{\max_i |x_i-y_i|^\infty}$$

$$\max_i |x_i-y_i|$$ ✓ That's it! Just the largest set of numbers in the absolute value → "the biggest"

When would you use this?

Suppose a glitchy machine needs to move a distance x and y. The time t to achieve this could be $x+y$, or if movements x and y move simultaneously then we can apply the **∞-norm distance**: time at x and y. In other words, time t to move would be whichever time is longer for x or y → $\max(x, y)$ time



k-means algorithm

Clustering
Grouping data points
Solve using k-means algorithm

x_{ij} = attribute j of data point i

$$y_{ik} = \begin{cases} 1, \text{ if data point i is in cluster k} \\ 0, \text{ if not} \end{cases}$$

z_{jk} = coordinate j of cluster center k

What we'd like to find is a set of k cluster centers and assignments of each data point to a cluster center to minimize the total distance from each data point to its cluster center. This piece adds up all of the distances from data points to cluster centers but only when the data point is in that cluster.

Goal: find a set of k cluster centers and (arbitrary) coordinates of each data point to a center cluster... all to minimize the total distance of each data point to its cent cluster.

$$\underset{y,z}{\text{Minimize}} \sum_i \sum_k \sqrt{\sum_j (x_{ij}-z_{jk})^2}$$

This adds the distances from data points to cent clusters, but only when the data point is in that cluster

$$\text{Subject to } \sum_k y_{ik} = 1 \text{ for each } i$$

Problem! This is a hard optimization problem to solve...

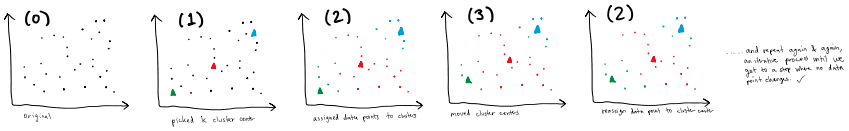
So instead, we use the **k-means algorithm**:

(1) Decide how many clusters we want the algorithm to give us by picking k cluster centers within range of data. (= the points we pick)
(2) Assign each data point to the nearest cluster center.
(3) But the points we called cluster points aren't really at the center of their clusters, so recalculate cluster centers. (centroids)
.... Repeat, starting with step 2, until no changes

(2) Assign each data point to the nearest cluster center.

(3) But the points we called cluster points aren't really at the center of their clusters, so recalculate cluster centers. (centroids)

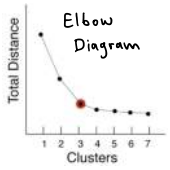
.... Repeat, starting with step 2, until no changes



Note that the k-means algorithm is:

- a heuristic → it's fast and it works, but it's not guaranteed to find the absolute best solution .... but it comes close!
- an expectation-maximization algorithm .... see steps 1-3 above.
  - Expectation: calculating center clusters is taking the mean of all the points in the cluster
  - Maximization: reassigning data points to cluster centers
    - minimizing: finding the smallest distance to a cluster center = maximizing the negative of the distance to a cluster center
  - Takes turns between expectation, maximizing, expectation, maximizing, etc → hence, "EM" algorithm

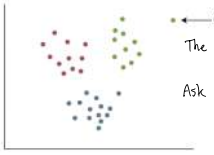
Since k-means algorithm = heuristic = fast, we can run it several times to determine how many clusters centers k to use, comparing the results to determine which to pick.



Theoretically, k could equal # of data points...

But this graph can give us a starting point of which k to use without blindly guessing.

What to do with outliers ...



Outlier = data point that's not close to any of the cluster centers

The k-means algorithm will assign the outlier to whichever cluster center is closest.... but as you can see in visual, this isn't always right.

Ask yourself the following:

What is this outlier?

What makes its attribute so different?

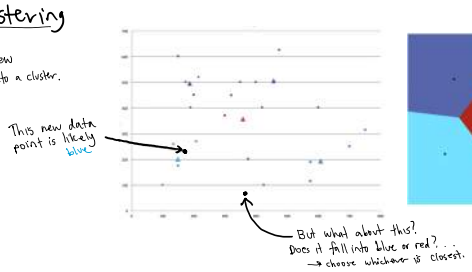
What's the implication of putting a point like this in the nearest cluster?

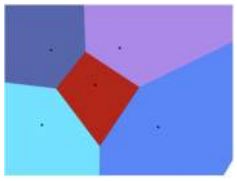
Understand the situation! Look deep at the data, algorithm, & situation to find the right answer.

Predictive Clustering

Based on previous clusters, new data points can be "predicted" into a cluster.

For example ...

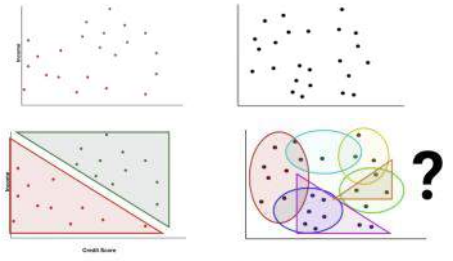




Can color-code the cluster ranges so we know where each data point could fall.

This is called the **Voronoi Diagram**.

Classification vs. Clustering



Correct classification of data points is already known

Know ✓ Attribute

Know ✓ Response

Correct classification of data points is not known

Know ✓ Attribute

NOT know ✗ Response

Supervised Learning: (classification)

machine learning where the "correct" answer/response is known for each data point in the training set → more common

Unsupervised Learning: (clustering)

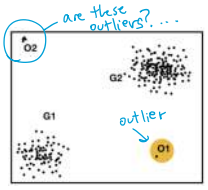
machine learning where the "correct" answer/response is NOT known for each data point in the training set

Week 3: Basic Data Preparation, Change Detection

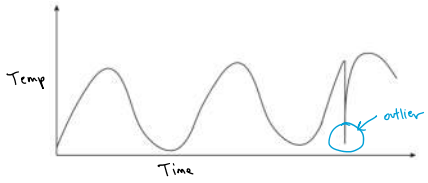
Tuesday, February 1, 2022 4:02 PM

How to Prepare a set of data for use in analytics.

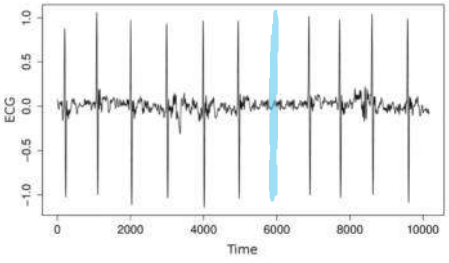
Data Preparation → process data before building model .... aka address problems in data beforehand
i.e. scale data, look for outliers, etc.



Outlier: data point that's very different from the rest

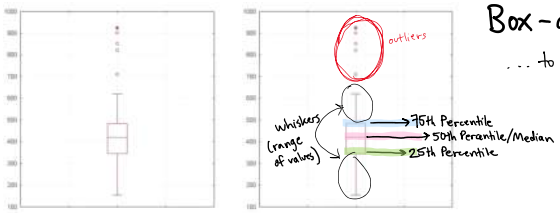


Contextual Outlier: $f(t) = y$ value isn't an outlier since it's not far from other points overall, but the value at time t is an outlier since it's far from points nearby in time



Collective Outlier: outlier by omission; something is missing in a range of points, but can't tell exactly where

Noticing outliers is easy. It's figuring out what to do with them that's hard.



Box-and-whisker plot
... to find outliers in one dimension

Unfortunately, there's no great way to detect multi-dimensional outliers.
However!
We can build a model, fit the parameters, then see which points have ↑ error.

For example ....
Say we did exponential smoothing here.
At that outlier point, the model's error will be very high.

Outliers: real, fake, or random?
Sometimes, the outliers are simply bad data. Other times, something is wrong with the situation/data set.
But most of all, it could just be randomness. 4% of data outside two standard deviations are expected to be outliers.

Consider context to outliers. There is no one right answer.

Change Detection → determining whether something has changed (usually time series data)

Why is this needed? → • determine if action might be needed i.e. time for preventive maintenance? global temperature change? decreased sales?
• determine impact of past action i.e. policy change have desired effect? with lower tax rates, were more people hired? did discount affect sales?

Why is this needed? →
- determine if action might be needed i.e. time for preventive maintenance? global temperature change? decreased sales?
- determine impact of past action i.e. policy change have desired effect? with lower tax rates, were more people hired? did discount affect sales?
- determine changes to help plan i.e. have voting patterns changed?

Case Studies

Manufacturing Plant: can predict when there is about to be a bottleneck, to prevent bottlenecks

Railroad Safety: to prevent train derailment, temperature sensors are along the track to predict failure (derailment). IF it's too high, the train will stop before derailment, do maintenance, then resume.

Problem: Sometimes there are outliers (random fluctuations) ... we wouldn't want to stop the train if it's not going to fail.

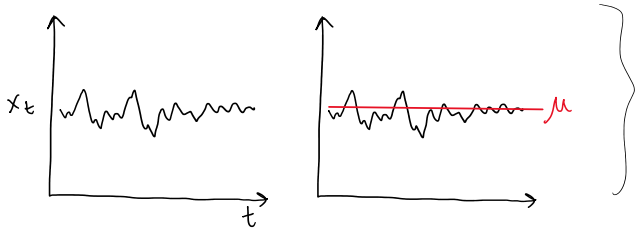
So, how do we determine if there's actually a change? Need something faster than hypothesis testing ....

That's where CUSUM comes in.

Cumulative Sum (CUSUM)

CUSUM detects increases, decreases, or both.

Answers the question: has the mean of the observed distribution gone beyond a critical level?



Suppose we have some data, X_t.
X_t = observed value at time t ... i.e. stock $ @ time t
μ = mean/expected value of observations x_t if there is no change.

Therefore,
$(X_t - \mu)$ = how much above expected the observation is at time t

CUSUM Metric: Calculate S_t and declare we've observed a change if S_t goes above threshold T.

$$S_t = \max\{0,\ S_{t-1} + (X_t - \mu)\}$$

(0: choose zero if $S_{t-1} + x_t - \mu$ is negative; S_{t-1}: adding to the previous value; $(X_t - \mu)$: amount above μ baseline line)

Is $S_t \geq T$?

(T: we pick threshold T)

Basically, if the running total would be below zero, it's irrelevant to the question of whether we later see an increase

HOWEVER!

Sometimes there is randomness. Sometimes, X_t will ~randomly~ be higher than the expectation.

How do we account for this? ...

$$S_t = \max\{0,\ S_{t-1} + (X_t - \mu - C)\}$$

(C: sensitivity value to account for randomness (like standard deviation))

Is $S_t \geq T$?

↑C, ↓S_t, ↑sensitive

or

↓C, ↑S_t, ↓sensitive

↑T, ↓time to detect changes, ↓false changes

or

↑T, ↑time to detect changes, ↑false changes

Detecting an increase

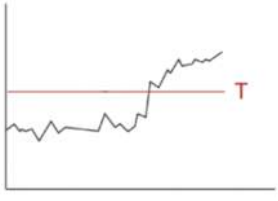
$$S_t = \max\{0,\ S_{t-1} + (X_t - \mu - C)\}$$

Is $S_t \geq T$?

Detecting a decrease

$$S_t = \max\{0,\ S_{t-1} + (-X_t + \mu - C)\}$$

Is $S_t \geq T$?



model parameters

How do you select C & T? (C & T: model parameters)

Model needs to find changes quickly, but not be so sensitive that it says there's a change when there isn't one.

But the parameters are totally dependent on risk tolerance of the situation.

...

Play around with T until it gets expected results?

C can be 2-3 standard deviations, but it still totally depends.

Week 4: Time Series Models ⟶ Exponential Smoothing, ARIMA, GARCH

Thursday, March 3, 2022 2:50 PM

Time Series Data: data where the same response is known for many time periods

Still, there could be: trends over time, cyclical variations, and randomness

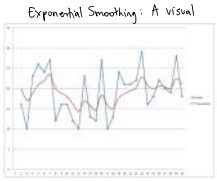
Say there's a patient getting a blood pressure reading and it comes out higher than average
.... But this could be random fluctuation. Should the doctor take action?
Without knowing a baseline value, it's hard to know what to do. This is where **exponential smoothing** comes in handy

Let's go back to the blood pressure scenario.

Say:
- S_t = expected baseline response at time period t ⟶ blood pressure at hour t
- X_t = the observed response ⟶ observed blood pressure at hour t

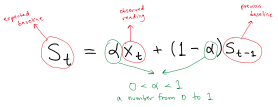
2 possibilities. Either:
- $S_t = X_t$ ⇒ the observed value (blood pressure) is the real indicator of the baseline .... as in, blood pressure is always that high.
- $S_t = S_{t-1}$ ⇒ today's baseline is the same as yesterday's baseline ..... as in, blood pressure is unusually high

Exponential Smoothing: A visual



Exponential Smoothing Method combines these two ideas.

Other names:
Single/Double/Triple Exponential Smoothing
(controls for: level, trend, seasonality)
Also: - Winter's Method
- Holt-Winters



[single]

$$S_t = \alpha X_t + (1-\alpha) S_{t-1}$$

Initial condition: $S_1 = X_1$

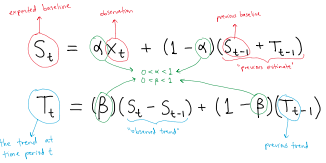
We choose α;
$0 < \alpha < 1$
⇒ $\alpha \to 0$: a lot of randomness ⟶ trusting S_{t-1} (previous estimate)
⇒ $\alpha \to 1$: little randomness ⟶ trusting X_t (observed value/what you see)

NOTE
does not work with trends or cyclical variations since the baseline is a constant

Note that every past observation contributes to the current baseline estimate.
It's just baked in the S_{t-1} equation.
⟶ But, newer observations are more important/has more weight.

What if there's a trend? The value we're measuring may naturally be increasing or decreasing over time.

Use this equation instead:



[trend/double]

$$S_t = \alpha X_t + (1-\alpha)(S_{t-1} + T_{t-1})$$
$$T_t = (\beta)(S_t - S_{t-1}) + (1-\beta)(T_{t-1})$$

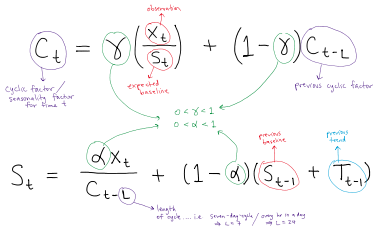
Initial condition: $T_1 = 0$

We choose α, β:
$0 < \alpha < 1$
$0 < \beta < 1$

What if there's a cyclic pattern? Trends may go up and down, but they repeat over a cycle/set length of time L.

You could add up trends in an additive way, or use equation below
Cyclic pattern = Seasonalities in a multiplicative way

Use this equation instead:



[cyclical/multiplicative seasonality/triple]

$$S_t = \frac{\alpha X_t}{C_{t-L}} + (1-\alpha)(S_{t-1} + T_{t-1})$$
$$C_t = \gamma\left(\frac{X_t}{S_t}\right) + (1-\gamma) C_{t-L}$$

Initial condition: first L values of C are set to 1
... since $C=1$ equals one cycle, similar to $T_1 = 0$ for trend

C_t can be used to inflate or deflate the value.
ex) if $C_t = 1.1$, at $t=7$ (Sunday) then sales were 10% higher just b/c it was Sunday
Make sure to use L from time periods ago. Here, C_{t-L}
ex) if today is Sunday use last Sunday's cyclic factor

We choose α, γ:
$0 < \alpha < 1$
$0 < \gamma < 1$

Forecasting

Exponential smoothing can analyze what happened in the past, but it can also predict what will happen next.

Basic exponential smoothing equation: $S_t = \alpha X_t + (1-\alpha) S_{t-1}$

Prediction: $S_{t+1} = \alpha X_{t+1} + (1-\alpha) S_{t+1}$

X_{t+1} is unknown ....
so, best guess: $X_{t+1} = S_t$ ⇒ "I predict the next observed value (X_{t+1}) will be the previous baseline estimate (S_t (instead of S_{t+1}))"

Best Guess
$X_{t+1} = S_t$

Therefore, forecast for time period $t+1$:

$$S_{t+1} = \alpha(X_{t+1}) + (1-\alpha) S_t \quad (X_{t+1} \to S_t)$$
$$S_{t+1} = F_{t+1} = \alpha S_t + (1-\alpha) S_t$$

Let $\alpha = 0$ ....

$$\hookrightarrow F_{t+1} = \cancel{(0)(S_t)} + \cancel{(1-0)} S_t$$

[single – forecasting]

$$F_{t+1} = S_t$$
$$F_{t+k} = S_t, \quad k = 1, 2, 3 \ldots$$

Trend Exponential Smoothing Equation:

Eqn ⇒ $S_t = \alpha X_t + (1-\alpha)(S_{t-1} + T_{t-1})$

Add +1 to t ⇒ $S_{t+1} = \alpha(X_{t+1}) + (1-\alpha)(S_t + T_t)$

(X_{t+1} is the equal S_t again)

Best Guess
ex.

$$= \alpha S_t + (1-\alpha)(S_t + T_t)$$

[double/trends – forecasting]

$$F_{t+1} = S_t + T_t$$

Best Guess
$X_{t+1} = S_t$

Eqn $\Rightarrow S_t = \alpha X_t + (1-\alpha)(S_{t-1} + T_{t-1})$

Add +1 to t $\Rightarrow S_{t+1} = \alpha X_{t+1} + (1-\alpha)(S_t + T_t)$

↳ by the equal St again

Simplify, with $X_{t+1} = S_t$:
$= \alpha S_t + (1-\alpha)(S_t + T_t)$
$= \alpha S_t + S_t + T_t - \alpha S_t - \alpha T_t$
$= S_t + T_t - \alpha T_t$

..... Let $\alpha = 0$ ....

$S_{t+1} = F_{t+1} = S_t + T_t$

[double/trends – forecasting]

$F_{t+1} = S_t + T_t$

$F_{t+k} = S_t + kT_t$, $k = 1, 2, 3 \ldots$

Multiplicative Seasonality Exponential Smoothing Equation:

$C_t = \gamma\left(\frac{X_t}{S_t}\right) + (1-\gamma)C_{t-L}$

$C_{t+1} = \gamma\left(\frac{X_{t+1}}{S_{t+1}}\right) + (1-\gamma)C_{(t+1)-L}$

.... Let $\gamma = 0$ .....

$\Rightarrow C_{t+1} = C_{t+1-L}$

$\Rightarrow S_t = \frac{\alpha X_t}{C_{t-L}} + (1-\alpha)(S_{t-1} + T_{t-1})$

$\Rightarrow S_{t+1} = \frac{\alpha X_{t+1}}{C_{t+1-L}} + (1-\alpha)(S_t + T_t)$

$= \frac{\alpha S_t}{C_{t+1-L}} + (1-\alpha)(S_t + T_t)$

..... Let $\alpha = 0$ .....

$\Rightarrow S_{t+1} = F_{t+1} = \frac{(0)S_t}{C_{t+1-L}} + (1-0)(S_t + T_t)$

$F_{t+1} = S_t + T_t$???

[triple/cyclical – forecasting]

$F_{t+1} = (S_t + T_t)\, C_{(t+1)-L}$

$F_{t+k} = (S_t + kT_t)\, C_{(t+1)-L(k-1)}$, $k = 1, 2, 3 \ldots$

Which α, β, and γ do I use?

Use Optimization! Use sum of squared error equation: $(F_t - X_t)^2$

Use the α, β, γ values in which error is minimized.

Exponential smoothing is for analyzing time series data where the same response is known for many time periods.

What if I want something more general?

ARIMA: AutoRegressive Integrated Moving Average

↳ Estimates or forecasts a value.

- A more general way to analyze time series data.
- Short-term forecasting... when data is more stable with fewer peaks, valley, & outliers
- Need 40 past data points for ARIMA to work well

3 parts: (1) Differences, (2) Autoregression, (3) Moving Parts

(1) Differences

We know that the exponential smoothing equation is:

$S_t = \alpha X_t + (1-\alpha) S_{t-1}$

$\Rightarrow S_t = \alpha X_t + (1-\alpha)\alpha X_{t-1} + (1-\alpha)^2 \alpha X_{t-2} + (1-\alpha)^3 \alpha X_{t-3} + \ldots + (1-\alpha)^n \alpha X_{t-n}$

This works well if the data is stationary, meaning that the mean, variance, and other measures are all expected to be constant over time.

..... But what if it's not stationary? i.e. has trend or seasonality

Often, data is not stationary.... but the differences are.

First-order difference $D_{(1)}$: difference of consecutive observations
$D_{(1)t} = (x_t - x_{t-1})$
Second-order difference $D_{(2)}$: differences of the differences
$D_{(2)t} = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2})$
Third-order $D_{(3)}$: differences of the differences of the differences
$D_{(3)t} = [(x_t - x_{t-1}) - (x_{t-1} - x_{t-2})] - [(x_{t-1} - x_{t-2}) - (x_{t-2} - x_{t-3})]$

↳ dth-order differences ⇒ $D_{(d)}$

(2) Autoregression

Autoregression predicts the current value based on previous time periods' values.

- Using earlier values to predict ..... this only works with time series data
- predicting the value based on other factors

Exponential Smoothing is an order-∞ autoregressive model.
It uses data all the way back.

⟶ order-p autoregressive models only go back p time periods.

⟹ ARIMA combines autoregression and differencing ⟶ autoregression on the differences
Use p time periods of previous observations
to predict d-th order time differences.

(3) Moving Average

Previous errors ε_t as predictors: $\varepsilon_t = \hat{x}_t - x_t$
If we go back q time periods, it's called an order-q moving average.

ARIMA(p,d,q) Model

$$D_{(d)t} = \mu + \sum_{i=1}^{p} \alpha_i D_{(d)t-i} - \sum_{i=1}^{q} \theta_i(\hat{x}_{t-i} - x_{t-i})$$

dth order differences
pth-order autoregression
qth-order moving average

Statistical software can be used to find d, p, q.
But also, here are some common ARIMA(p,d,q) functions:

- ARIMA(0,0,0): white noise (no patterns)
- ARIMA(0,1,0): "random walk"
- ARIMA(p,0,0): AR (Autoregressive) model
- ARIMA(0,0,q): MA (Moving Average) model
- ARIMA(0,1,1): basic exponential smoothing model

ARIMA is great for estimating or forecasting a value.
But what if you want to estimate or forecast a variance? ⟶ use GARCH

GARCH: Generalized Autoregressive Conditional Heteroskedasticity

↳ Estimate or forecasts the variance.

Why would we need to know the variance?

⟹ variance = how much forecast might be higher or lower than the true value
So, ⟹ using GARCH can help us estimate the amount of error

Variance estimation is important in investment (portfolio optimization model)
↳ risk vs. returns
↳ Variance: a proxy for the amount of volatility or risk

GARCH (p,q) Model

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \gamma_i \epsilon_{t-i}^2$$

Similar to ARIMA, but with key differences:

ARIMA	GARCH
• Observations	• Variances
• Linear Errors	• Squared Errors
• Differences of Variances	• Raw Variances
• Has d value: ARIMA(p,d,q)	• No d value: GARCH(p,q)

[summary] this too

GARCH

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \gamma_i \epsilon_{t-i}^2$$

ARIMA

$$D_{(d)t} = \mu + \sum_{i=1}^{p} \alpha_i D_{(d)t-i} - \sum_{i=1}^{q} \theta_i (\hat{x}_{t-i} - x_{t-i})$$

Week 5: Basic Regression

Friday, March 4, 2022 3:42 PM

Regression is a way to determine if a variable has an impact on the system.

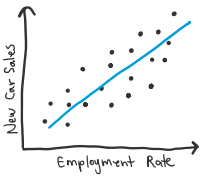
Regression can:
(1) Answer questions about how systems work (descriptive)
(2) Make predictions about what will happen in the future (predictive)

EXAMPLES
(1) how many runs the average home run is worth in baseball
(1) estimate the effect of economic factors on presidential election voting
(1) find the importance of a general education on their future income
(1) discover key factors in purchasing a specific type of car
(2) predict how tall a child will be as an adult
(2) forecast the price of oil a year into the future
(2) predict the remaining lifetime of inventory applying the LIFO assumption
(2) predict demand of housing in a city over the next six months

Simple Linear Regression (SLR)

↳ Linear Regression with one predictor.
Looking for a linear relationship between the predictors and the response.

Let's look at an example....



y = response ⟶ new car sales
X_1 = predictor ⟶ employment rate

So, when more people are working, there are more people who can afford to buy a new car, so new car sales are higher... But how much higher?? That's what regression will find out.

Regression Equation (with one predictor): $y = a_0 + a_1 X_1$ } a = attribute = predictor

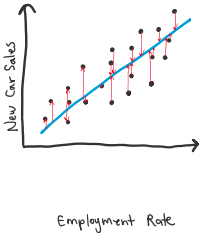
But what if we had m predictors? ....

Regression Equation with m predictors:

$$y = a_0 + a_1 X_1 + a_2 X_2 + \dots + a_m X_m$$
$$\Rightarrow y = a_0 + \sum_{j=1}^{m} a_j X_j$$

Is **this** a good line? ....

⟹ Find the sum of squared errors to measure the quality of the line's fit.



y_i = cars sold at data point i

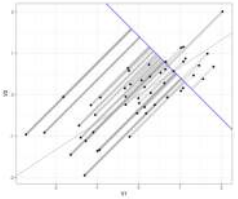
$\hat{y}_i$ = model's prediction of cars sold (@ regression line of $f(x)$)

Data point i prediction error: $y_i - \hat{y}_i = y_i - (a_0 + a_1 X_{i1})$

$\Rightarrow \hat{y}_i = a_0 + a_1 X_{i1}$

Sum of Squared Errors: $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Gif showing various SSE to line fit.



Open on desktop to view :)

Minimize SSE to find the best Regression Line ⟶

$$SSE = \sum_{i=1}^{n} \left[y_i - (a_0 + a_1 X_{i1}) \right]^2$$
$$= \sum_{i=1}^{n} \left[y_i - (a_0 + \textstyle\sum_{j=1}^{m} a_j X_{ij}) \right]^2$$

Underlying math: SSE is a convex function we're trying to minimize.
Take partial derivatives of the SSE w.r.t each constant
Set each partial derivative to zero, then solve those equations simultaneously to find the minimized SSE, aka best fit solution ⟶ or just use software ☺

How do we measure the quality of a model's fit?

A few different ways. Examples: Likelihood, AIC, BIC

Likelihood: measures the probability (density) for any parameter set

We assume that the observed data is the correct value, and that we have information about the variance.

Maximum Likelihood: parameters that give the highest probability ⟶ aka, best fit set of parameters

ex) Error $\sim N(0, \sigma^2)$, i.i.d. (distributed, normal, mean, variance, independent and identically distributed)
Observations: $z_1, \dots, z_n$ (previously y_i)
Model estimates: $y_1, \dots, y_n$ (previously $\hat{y}_i$)

Suppose we believe that errors are normally distributed with mean zero and variance σ^2, and they're independent from one data point to the next.

Then, given observed data z_1 through z_n, and our model estimates y_1 through y_n, below is the probability of observing z_i if the true value is y_i.

ex) Error ~ $N(0, \sigma^2)$, i.i.d.
Observations: $z_1, \ldots, z_n$ (previously y_i)
Model estimates: $y_1, \ldots, y_n$ (previously $\hat{y}_i$)

… normally distributed with mean zero and variance σ^2, and they're independent from one data point to the next.

… estimates y_1 through y_n, below is the probability of observing z_i if the true value is y_i.

Probability density for observing z_i if true value is y_i $= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)\left(e^{\frac{-(z_i - y_i)^2}{2\sigma^2}}\right)$

since independent? $\Rightarrow \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \left[e^{\left(\frac{-1}{2\sigma^2}\right)\left(\sum_{i=1}^n (z_i - y_i)^2\right)}\right]$

Can find largest value by finding n

Simplifying this gets us MLE
$\Rightarrow$ Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE):

the set of parameters that minimizes the sum of squared errors

$$maximize \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n e^{-\frac{1}{2\sigma^2}\sum_{i=1}^n (z_i - y_i)^2} \rightarrow \left[minimize \sum_{i=1}^n (z_i - y_i)^2\right]$$

Therefore, in regression:

$$SSE = \sum_{i=1}^n \left[y_i - \left(a_0 + \sum_{j=1}^m a_j x_{ij}\right)\right]^2$$

(y_i → z_i)

Remember, this assumes model is independent with normally distributed errors

$$MLE = minimize \sum_{i=1}^n \left[z_i - \left(a_0 + \sum_{j=1}^m a_j x_{ij}\right)\right]^2$$

…. y_i & z_i are the same btw

Note that we can compare two different models by using the likelihood ratio (ratio of their likelihood), then conduct a hypothesis test.

Some more complex approaches….

Akaike Information Center (AIC):

Note
↑ parameters can improve a model… but also lead to overfitting. It pushes the model to fit random effects, not the real ones.

$$\Rightarrow AIC = 2k - 2\ln(L^*)$$

L^* = maximum likelihood value
k = number of parameters being tested (attributes)

"Penalty Term" ($2k$)
↳ Balances likelihood with simplicity to help prevent overfitting.

$$AIC = 2k - 2\ln(L^*)$$

Now to substitute this into regression.

$$AIC = 2k - 2\ln(L^*)$$

(k: $m+1$, for a_0 thru a_n; L^*: sub in likelihood function)

$$\Rightarrow AIC = 2(m+1) - 2\ln\left(\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n e^{-\frac{1}{2\sigma^2}\sum_{i=1}^n \left(z_i - \left(a_0 + \sum_{j=1}^m a_j x_{ij}\right)\right)^2}\right)$$

↓AIC is preferred; results in ↓k and ↑L^*

Note that AIC has nice properties, if there are infinitely many data points. Obviously this is impossible …. so, for "smaller" data sets, use Corrected AIC (AIC_c)

Corrected AIC (AIC_c):

we generally use this unless we have an extremely large (→∞) data set

$$\Rightarrow AIC_c = AIC + \frac{2k(k+1)}{n-k-1} \Rightarrow AIC_c = 2k - 2\ln(L^*) + \frac{2k(k+1)}{n-k-1}$$

When comparing models with different AIC_c/AIC values, we can calculate the relative probability that one of them is better than the other.

When comparing models with different AIC_c/AIC values, we can calculate the relative probability that one of them is better than the other.

This is called Relative Likelihood.

$$\text{Relative Likelihood} = e^{\frac{AIC_1 - AIC_2}{2}}$$

Example:
Model 1: AIC = 75
Model 2: AIC = 80
⇒ Relative Likelihood = $e^{\frac{(75-80)}{2}} = 8.2\%$
.... Model 2 is 8.2% as likely as Model 1 to be better... that's pretty low.
So, Model 1 is probably better and thus preferred.
↳ we knew this already since AIC_1 is smaller...?

But what if you have a lot more data than parameters?
Use Bayesian Information Criterion (BIC).

Bayesian Information Criterion (BIC): similar to AIC, but encourages models with fewer parameters (attributes)

$$AIC = 2k - 2\ln(L^*)$$
$$\Rightarrow BIC = k\ln(n) - 2\ln(L^*)$$

("penalty term")

Note that BIC's penalty term > AIC's penalty term....

L^* = maximum likelihood value
k = number of parameters being tested (attributes)
n = number of data points

$$BIC = k\ln(n) - 2\ln(L^*)$$

so, only use BIC when there are more data points than parameters.

When comparing Model 1 BIC (BIC_1) and Model 2 BIC (BIC_2)....

- $0 < |BIC_1 - BIC_2| < 2 \Rightarrow$ Smaller-BIC model is "slightly likely" better
- $2 < |BIC_1 - BIC_2| < 6 \Rightarrow$ Smaller-BIC model is "somewhat likely" better
- $6 < |BIC_1 - BIC_2| < 10 \Rightarrow$ Smaller-BIC model is "likely" better
- $|BIC_1 - BIC_2| > 10 \Rightarrow$ Smaller-BIC model is "very likely" better

Final Notes: There is no rule to pick AIC (frequentist POV) over BIC (Bayesian POV) or MLE.
All three can give valuable information,
and using all three can help you decide which model is better.

Interpreting Regression Coefficients

Components of Analytics:

- Descriptive Analytics
- Predictive Analytics

(Regression can do these two, hence why it's so common.)

- Prescriptive Analytics

If we're using descriptive regression to figure out how the system works, the key is to look at the coefficients.

ex) Baseball example: want to determine average number of runs a homerun is worth

Response: how many runs a team scored that season

Predictors: Number of homeruns, triples, doubles, singles, outs, double plays, stolen bases, etc.

Regression Model:

$$\text{Runs Scored} = a_0 + a_1[\text{number of homeruns}] + a_2[\text{number of triples}] + \dots + a_7[\text{number of stolen bases}]$$

↳ Let's say this is 1.4.
So, every home run hit will add 1.4 runs to the team's total

If we're using predictive regression to make forecasts, the key is to look at the predicted responses.

ex) Height Example: want to predict how tall a 2-year-old will be as an adult

Response: a person's height as an adult

Predictors: father's height, mother's height, age 2 height, male/female

Regression Model:

$$\underset{\text{(cm)}}{\text{Adult Height}} = a_0 + a_1[\text{father's height}] + a_2[\text{mother's height}] + \dots + a_4[\text{male or female}]$$

→ Let's say this is 1.4.
So, every cm of the father's height will add 1.4 cm to the predicted adult height.

Causation vs. Correlation

Causation: one thing causes another thing
Correlation: two things tend to happen or not happen together → Neither of them might cause the other.

Correlation does NOT equal causation!

ex)

y: hours/day spent outdoors in winter x_1: city's avg. daily winter temperature $y = a_0 + a_1 x_1$	y: hours/day spent outdoors in winter x_1: city's avg daily winter temperature $x_1 = b_0 + b_1 y$
Correlation between y and x_1 • Low p-value of a_1	Same correlation between y and x_1 • p-value of b_1 = p-value of a_1
Does higher winter temperature cause people to go outside?	Does people spending more time outdoors in the winter cause higher winter temperatures?
Kind of (probably)	That's pretty silly!

There's a correlation here between temperature and people who go outside, but we can't say one caused the other.

i.e.
✗ ↑ temperatures ⇒ ↑ people who go outside .... seems possible
✗ ↑ people who go outside ⇒ ↑ temperatures .... obviously wrong

How to tell when there is causation....

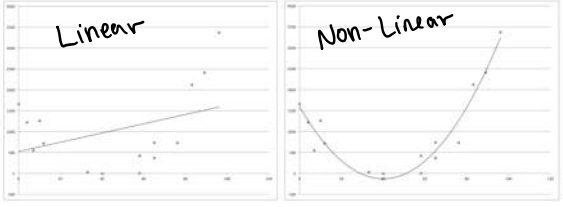
Actually very difficult; easier to prove that there is no causation.
Here are starting points, though:

- Cause is before the effect
- The idea of causation makes sense
- No outside factors causing the relationship → most important one, but the hardest. There could be infinite of factors....

In short: be careful before causing causation.

Generalized Regression Modeling

SLR is great for linear models....
But what if it isn't linear?



Transform the data.
We can adjust the data so the fit is linear.

Quadratic Regression: $y = a_0 + a_1 X_1 + a_2 X_1^2$ → add exponential

Variable Interaction: $y = a_0 + a_1 X_1 + a_2 X_2 + a_3 (X_1 X_2)$

Response Transform: $\log(y) = a_0 + a_1 X_1 + \dots + a_m X_m$

Can really transform anything....

ex) $y = a_0 + a_1 X_1^{1.5} + a_2 X_1^2 X_3 + a_3 \sin(X_2)$

$\ln(y) = a_0 + a_1 X_1^2 X_2 X_3 + a_2 \log(X_3)$

Outputs

So, now we know how to fit and build regression models.
Now it's time to look at outputs and determine which attributes are important, and which are not.

P-Values

(form of hypothesis testing)

..... estimates the probability that the coefficient a_m is equal to zero.

i.e. p-value = 0.02 $\Rightarrow$ 2% of not being significant
$\Rightarrow$ 98% of being significant

correlations:
- $\uparrow$ p-value (toward 1) $\Rightarrow$ attribute is not important
- $\downarrow$ p-value (toward 0) $\Rightarrow$ attribute is important

$\Rightarrow$ If p-value $> \underset{\alpha}{0.05}$, then remove the attribute from the model.

Note that other thresholds α can be used.
$\alpha = 0.05$ is pretty standard though.
- $\uparrow \alpha$, $\uparrow$ attributes, $\uparrow$ likelihood of irrelevant attribute
- $\downarrow \alpha$, $\downarrow$ attributes, $\downarrow$ likelihood of irrelevant attribute

Warning! As $\uparrow$ data, $\downarrow$ p-values.... even with irrelevant attributes.
Be careful to rely on p-values when there is a lot of data.

Confidence Interval

Most software will give a 95% confidence interval around that coefficient.
So, you can see where the coefficient probably lies, and how close that is to zero.

t-Statistic

..... the coefficient divided by its standard error
Similar to p-value. It's just another way of finding the same information.

Coefficient

Sometimes you'll discover that the coefficient doesn't make much of a difference, even when it has a low p-value, and even when multiplied by the attribute.

i.e. when looking at population predictions, an $a_n = 1$ will hardly make a difference.

R-squared Value (R^2)

..... estimates how much variability your model accounts for.

For example, an $R^2 = 59\%$ means the model has accounted for 59% of the variability in the data.
The 41% is from randomness or other factors not modeled.

Adjusted R-squared (R^2_{adj}): adjusts for the number of attributes used

When using R^2, keep in mind that some things are not easily modeled.

In homework, we may get R^2 values at the ~90% range.

This hardly reflects real life.

In social sciences, for example, R^2 values of 0.3 or 0.4 are quite good.

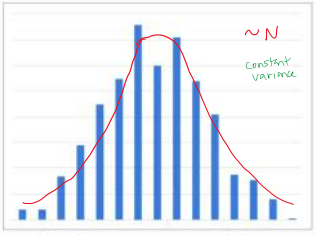
Week 6: Advanced Data Preparation

Saturday, March 5, 2022 12:47 PM

Sometimes, it's helpful to transform data before using it to fit a model.

For example....

Recall variance:
- $\downarrow \sigma^2 \Rightarrow$ data points are similar; do not vary widely from the mean μ
- $\uparrow \sigma^2 \Rightarrow$ data points are not similar; varies widely from the mean μ

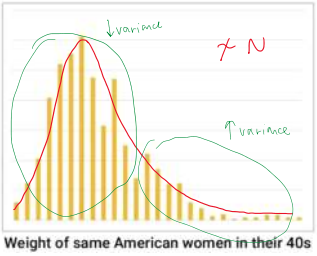


Height of American women in their 40s

Let's say we have a normality assumption.

This would be true for the graph on the left.

Normality is a pretty good fit to the normal distribution, so the variance is constant across the whole distribution.



Weight of same American women in their 40s

But not all data has normality.

Results will have bias when normality assumption is wrong.

The graph on the left has varying amount of variance, which can introduce bias.

Heteroscedasticity: unequal / differences in variance

So, what can we do?

How do we deal with heteroscedasticity?

↳ Transform the data to fit the model, i.e. fit to the normal distribution

Box-Cox Transformation ..... transforms response to eliminate heteroscedasticity.

↳ A logarithmic transformation that:
- stretches out the smaller range to enlarge its variability,
- shrinks the larger range to reduce its variability

$$t(y) = \frac{(y^\lambda - 1)}{\lambda}$$

- $\lambda \Rightarrow$ value we're optimizing
- y = vector of responses
- $t(y)$ = transform vector

The above equation will result in being close to normal distribution.

When would we have to use the Box-Cox transformation?

↳ Check by using a Q-Q plot.

See Week 9 for more information

Detrending

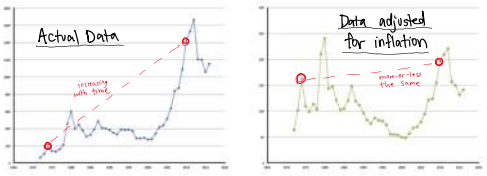
Sometimes trends in time series can influence/mess up a factor-based analysis.

So, we'd want to "detrend"/remove the trend from the data.

Consider detrending when doing factor-based modeling like regression and SVM.

For example...

The Price of Gold over the past 45 years



This detrends the response.
However, we can also detrend predictors.

Looking at 1974 and 2009 data, we can see that detrending gives two different values.

How to Detrend

In the case of the price of gold example, detrending is easy since there are historical inflation rates.
So, adjust each data point individually by the amount of historical inflation.

However, most cases aren't like this.
In most cases, we don't know what the trend has been in the past, except for our own data we're trying to detrend.

Solution → go factor-by-factor and fit a one-dimensional regression to it (could use more complicated function if you want)

one-dimensional regression: $y = a_0 + a_1 X$

ex) Simple linear regression for gold prices:

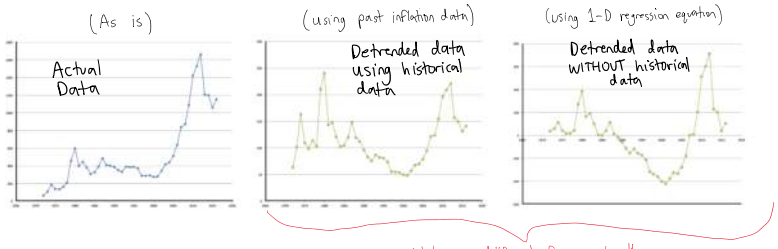
$\Rightarrow y = a_0 + a_1 x$

$\Rightarrow$ Price = -45,600 + (23.2)(year)

Detrended price:

$\Rightarrow y_{detrend} = a_{actual\ value} - y$

$\Rightarrow Price_{detrend} = Actual\ price - (-45{,}600 + (23.2)(year))$



Not so different from each other....
1-D regression difference comes from linear fit, which assumes a constant rate of inflation.
vs. inflation-adjusted data that uses different inflation numbers year-by-year.

Principal Component Analysis (PCA)

Used in high-dimensional and correlated data.
↳ PCA removes these correlations and ranks coordinates by importance

In other words,

A PCA plot converts the correlations (or lack thereof) of all the attributes into a 2D graph. Cells that are highly correlated are clustered together.

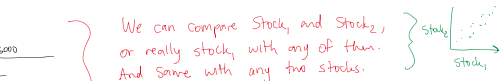
ex) Does the past performance of any particular stock indicate how well the overall market will perform the next day?

↳ Say there are 6000+ securities, and we have "normal" stock trade data.
The table could look like this:

↳ Say there are 6000+ securities, and we have "normal" stock trade data.
The table could look like this:

	$Stock_1$	$Stock_2$	$Stock_3$	$Stock_4$	$Stock_5$	.....	$Stock_{6000}$
t_1	~	~	~	~	~		~
t_2	~	~	~	~	~		~
t_3	~	~	~	~	~		~
t_4	~	~	~	~	~		~
t_5	~	~	~	~	~		~
⋮							
t_n	~	~	~	~	~		~

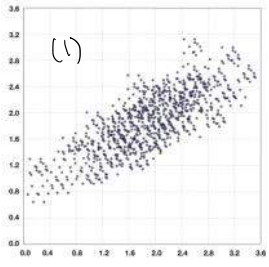
We can compare $Stock_1$ and $Stock_2$, or really stock, with any of them. And same with any two stocks. But how do we compare three stocks? Or four, or five, or six?. Can't make 6-D graphs, or a million 2D graphs....



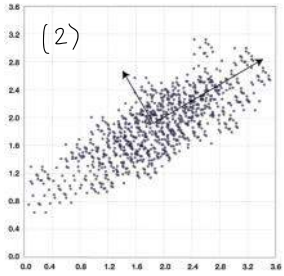
This is why PCA is useful.

Plus, underlying situations may have changed in time...
And there are high correlations between some of the predictors.
→ PCA can "reduce" the amount of data needed by addressing both of these issues.

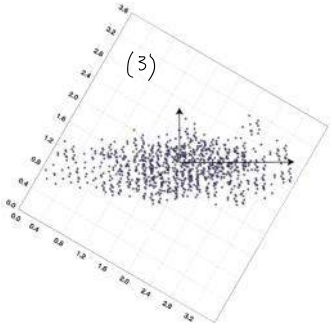
What does "removing correlation look like?: A visual



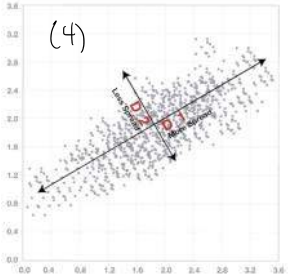
(1) There is clear correlation here....



(2) Marking the correlation



(3) If we pivot the graph, now the marked correlation line is at (0,0). So, the correlation is removed!
This is just to visually see what's going on. We can rotate it back and redefine a coordinate system.



(4) Let's redefine the coordinate system, then... into axes D_1 and D_2.
Notice that there is more spread among D_1 than D_2.
PCA will recognize that and automatically make D_1 the first dimension, and D_2 the second dimension.

In summary, PCA can:

⇒ Transform data
- Removes correlation between data
- Ranks coordinates by importance

⇒ Concentrate on the first n principal components
- Reduces effect on randomness
- Earlier principal components are likely to have higher signal-to-noise ratio

.... Basically, you start with a large number of factors, then use PCA to generate a small number of factors that you can use to fit the model.

Math of PCA

To go through math of PCA, you first need to understand eigenvalues/eigenvectors.

A refresher:

ex) Linear Algebra:

$$\begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & -3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 0 \end{pmatrix} = 3 \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

eigenvector (vector value) eigenvalue (scale value)

⇒ (magnitude)(direction)
⇒ (eigenvalue)(eigenvector)
i.e. ⇒ $3\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$

Linear transformation

$A\vec{v} \Rightarrow \lambda\vec{v}$ (eigenvalue, eigenvector)

vectors don't change directions, just magnitude.

Say we have a square matrix....
- $\vec{v}$ is a vector such that: $A\vec{v} = \lambda\vec{v}$
- $\vec{v}$ is an eigenvector of A
- λ is an eigenvalue of A : $\det(A - \lambda I) = 0$ (determinant)

Given λ, solve $A\vec{v} = \lambda\vec{v}$ to find eigenvalue $\vec{v}$.

⇒ Software can do this.

Now, back to PCA math.

Essentially, PCA uses the properties of eigenvectors and eigenvalues to get not just transformed set of coordinated directions, but also for the directions to be all orthogonal to each other.

Scaled matrix X of data:

- X: initial matrix of data
- X_{ij} = jth factor value for data point i, after scaling
- Scale such that $\frac{1}{m}\sum_i X_{ij} = \mu_j = 0$

Need to find eigenvectors $V_1 \dots V_n$ of (X^TX) { X transposed × X

- V: matrix of eigenvectors (sorted by smallest to largest eigenvalue)
- $V = [V_1 \; V_2 \dots]$, where V_j is the jth eigenvector of X^TX

Find the principal components....

- Multiply X by the eigenvectors
  - → First component is XV_1, second component is XV_2, etc.
- $XV_1, XV_2, \dots, XV_n$ are the principal components
  - → these are the set of orthogonal coordinate directions
- If a lot of data and just trying to remove correlation: use all PCs.
  But if you're trying to find a smaller number of variables in model, then pick n number of PCs for model.

This is all assuming linear transformation.

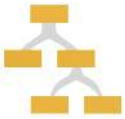
If nonlinear functions ⟶ use kernels.

Week 7: Advanced Regression

Sunday, March 6, 2022 3:15 PM

We've built, fitted, and evaluated regression models.
Now, let's divide data sets and have different models for each data subset.
We can do this with trees.

Classification and Regression Trees (CART)



This is a decision tree.

They can help us make a decision, and we can separate our models into different "systems" to our liking.

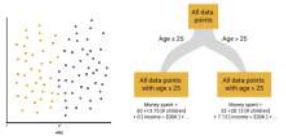
Example for a single regression model.....

Say we want to find the impact of a marketing email on recipient spending.

Predictors could include:
- Demographics: Age, sex, # children, income, etc
- Purchasing factors: avg spent per month on site
- Binary factors (was the email received? y/n)

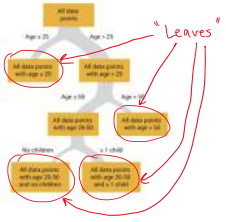
But what if some of those factors behave differently in different combinations?
i.e. maybe the coefficient for receiving a marketing email should really differ by age

We can split our model into two branches....



Regression for ≤ 25 year olds:
→ Money spent = 50 + 13.95[# of children] + 0[income ≥ $30k] + ...

Regression for > 25 year olds:
→ Money spent = 32 + 28.13[# of children] + 7.13[income ≥ $30k] + ...



It doesn't end here!
We can make as many branches as we want.
When a branch "ends," they're called leaves.

For each leaf, we'll run a separate regression to find the individual set of coefficients. { only do regression on the LEAF/leaves

The tree approach can:
- Be used as a flow chart to determine which model to use
- Can compare one leaf to another to see if the model needs to be improved, i.e. can compare R^2 values for each leaf

Note we can use decision trees for any model, not just regression!

So, how do we branch, and when do we stop branching?

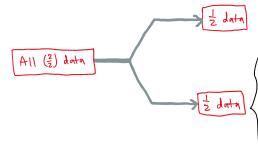
In theory, we could have a branch for every one data point in it.....
But that would be ridiculous and a very poor predictor.

Rule of thumb: each leaf contains at least 5% of the data

If the branch can lead to overfitting, reject it.

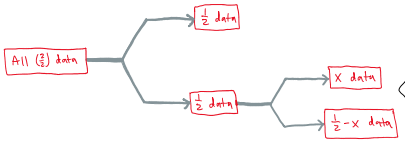
To begin branching, start with one factor at a time.

(1) —



Calculate total variance on this ½ data set
Now, if we remove one factor to split, how much lower would the total variance be?
Pick the factor that results in the lowest total variance.
If it's > Δthreshold difference, and >5% total data, make the split. ✓

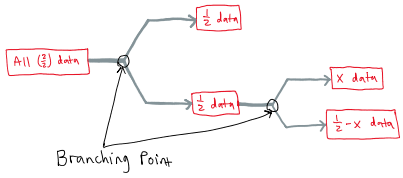
(2) —



Repeat above, until there is no split.

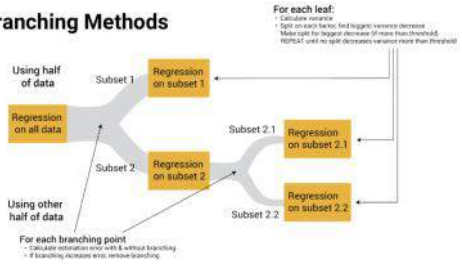
AKA, if:
- $\text{Variance}_{\text{split}} < \text{Variance}_{\text{total}}$
  - ↳ < Δthreshold if needed − Δvariance
- < 5% total data

Repeat for other ½ of data branch.



For each branching point:
- Calculate estimation error with and without branching.
- If branching increases error, remove branch.
  - ↳ "pruning"

Lecture's Flow Chart



Random Forests ..... Forests = many trees together

Random forests introduce randomness and generate many different trees with varying strengths and weaknesses.

Idea: overall, the average of all these trees is better than a single tree with specific strengths and weaknesses. → helps prevent overfitting

Steps to introduce randomness:

(1) give each tree slightly different datasets ("bootstrapping" process)
↳ tree could have multiple of the same points, and/or none of others

(2) randomly branch
↳ randomly choose a small number of factors X (instead of doing 1 factor at a time), and then choose the best factor with X to branch on
..... a common number of factors to use is $1 + \log(n)$

(3) don't prune the tree
↳ keep all the branches, even if error is high

We may end up with 500-1000 trees.... which one do we use?
↳ It depends.

- Regression Trees ⇒ use the average predicted response
- Classification Tree ⇒ use the mode, the most common predicted response

Unfortunately, random forests don't give us a specific regression/classification model from the data.
Still, used as a "default" model → if you want quick results and don't have a reason to try another method, use random forests.
But you may want to use another model if you need detailed insights to what's going on.

Confusion Matrices

Can measure how well a classification-type model works, since it can incorrectly/correctly classify something.

True Classification \ Model's Classification	Yes	No
Yes	Correct	Incorrect
No	Incorrect	Correct

⟹

True Classification \ Model's Classification	Yes	No
Yes	True Positive (TP)	False Negative (FN)
No	False Positive (FP)	True Negative (TN)

Guidelines
- True — model got it right
- False — model got it wrong
- Positive — model says it's in the category
- Negative — model says it's NOT in the category

Let's see an example.

ex) Say there is an email spam filter, and 50% of the incoming email is spam.

Confusion matrix:

True Classification \ Model's Classification	Real	Spam
Real	True Positive (TP) 490	False Negative (FN) 10
Spam	False Positive (FP) 100	True Negative (TN) 400

Question:

a) What fraction of the inbox will be spam with this filter?
⇒ total emails in inbox = TP + FP ⇒ 490 + 100
↳ fraction of spam in inbox = $\frac{FP}{TP+FP} = \frac{100}{490+100} \approx 17\%$ ⟶ 1 out of 6

b) What fraction of the inbox is real email flagged as spam?
⇒ total emails in inbox = TP + FP ⇒ 490 + 100
↳ fraction of lost emails = $\frac{FN}{TP+FP} = \frac{10}{490+100} \approx 2\%$

Some common equations ... .

more common ones:
- Sensitivity = $\frac{TP}{TP+FN}$ ⟹ fraction of category members correctly identified
- Specificity = $\frac{TN}{TN+FP}$ ⟹ fraction of non-category members correctly identified

Others that you don't need to memorize:

Evaluating a model's quality ..... using a confusion matrix

Let's go back to the spam example.

Question:

Evaluating a models quality..... using a confusion matrix

Let's go back to the spam example.

True Classification \ Model's Classification	Real	Spam
Real	True Positive (TP) 490	False Negative (FN) 10
Spam	False Positive (FP) 100	True Negative (TN) 400

Question:
what is the cost of lost productivity [reading spam]?

Given:
- $0 lost for correct classifications → TN, TP
- $0.04 lost to read spam → FP
- $1 lost when missing a real message → FN

a) If 50% of email is spam:

$$\Rightarrow \text{Total cost lost} = TP(\$0) + FN(\$1) + FP(\$0.04) + TN(\$0)$$
$$= (490)(\$0) + (10)(\$1) + (100)(\$0.04) + (400)(\$0)$$
$$= \$14$$
$$= \$0.014/\text{email} = 1.4¢/\text{email}$$

a) If 40% of email is spam:

↳ Problem: given data is for 50% rate.....
so, we need to scale our math.

..... 40% is spam, 60% is real.....

Spam gets scaled down from 50% to 40%;
Real email gets scaled up from 50% to 60%.

$$\Rightarrow \text{Total cost lost} = (490)\left(\tfrac{0.4}{0.5}\right)(\$0) + (10)\left(\tfrac{0.6}{0.5}\right)(\$1) + (100)\left(\tfrac{0.4}{0.5}\right)(\$0.04) + (400)\left(\tfrac{0.6}{0.5}\right)(\$0)$$
$$= \$15.2$$
$$= 1.52¢/\text{email}$$

Why is this useful?

So we reduce the original 17% inbox spam to 10%,
but in return, it means there are more FNs (real emails flagged as spam).

Is this really better? ....
With the equations above, we can calculate a higher cost of productivity lost.

↳ This is a great example of weighing different attributes. (see SVM notes)

Logistics Regression

We've worked with models that are continuous....

But what if we want to estimate a probability? ⟹ use logistic regression model

ex)
- probability that the loan recipient will repay the entire loan in time ⟹ answer is 0 through 1 (but can include 0 and 1) ⟹ logistic regression
- yes or no: the recipient will pay back the loan in time ⟹ answer is 0 or 1 ⟹ linear regression

Basically,
Linear regression is a straight line.
Logistic regression is a curvy line.

⇒ Standard Linear Regression Model:

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_jx_j$$

⇒ Logistic Regression Model:

$$p = \frac{1}{1 + \exp\left[-(a_0 + a_1x_1 + a_2x_2 + \ldots + a_jx_j)\right]}$$

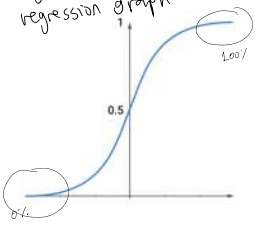
$$\log\left(\frac{p}{1-p}\right) = a_0 + a_1x_1 + a_2x_2 + \ldots + a_jx_j$$

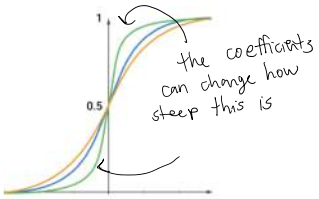
where:
p = probability of event you want to observe

↳ If:
- $a_0 + a_1x_1 + a_2x_2 + \ldots + a_jx_j = -\infty \Rightarrow p = 0$
- $a_0 + a_1x_1 + a_2x_2 + \ldots + a_jx_j = +\infty \Rightarrow p = 1$

Typical logistics regression graph

Typical logistics regression graph





Logistic Regression vs. Linear Regression

Similarities:

- Transformations of input data
- Consider interaction terms
- Variable selection
- Logistic regression trees
- Random logistic regression forests

Differences:

- (Relatively) Longer to calculate
- No closed-form solution
- Understanding model quality

→ Linear regression has an R^2 value (= fraction of variance explained by model)

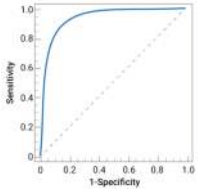
.... Logistics regression doesn't have one.

But, people made one up that sort of works, called the "pseudo R^2 value" to compare different models
Just keep in mind that it doesn't measure the same thing (variance)

Thresholding

Answer "yes" if probability p is at least some number; otherwise, answer "no."
i.e. If $p \geq 0.7$, give loan. Otherwise, no.

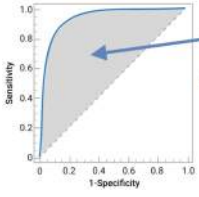
A visual way of looking at this is the Receiver Operating Characteristic (ROC) Curve.
It plots the (1-specificity) and (sensitivity).
x-axis y-axis



Sensitivity = (True Positives) / (True Positives + False Negatives)

Specificity = (True Negatives) / (True Negatives + False Positives)

See the Lesson (10.5) on Confusion Matrices



Area Under Curve

AUC = Probability that the model estimates a random "yes" point higher than a random "no" point

ex) Say Joe repaid the loan, and Moe did not repay the loan

⇒ AUC = probability that the model gives Joe's data point a higher response value than Moe's

For reference: AUC = 0.5 ⇒ basically just guessing

Advanced Topics in Regression

- Poisson Regression
  Use when response follows a Poisson distribution. $f(z) = \frac{\lambda^z e^{-\lambda}}{z!}$
- Regression Spline
  Use when you want to fit different functions to different parts of the data set; smooth connections between parts
- Bayesian Regression
  Use when combining expert opinion/broad distribution with small amount of data

- **Regression spline**
  Use when you want to fit different functions to different parts of the data set; smooth connections between parts

- **Bayesian Regression**
  Use when combining expert opinion/broad distribution with small amount of data

- **k-Nearest Neighbor (KNN) Regression**
  Use when needing to predict a response; we average response of k closest data points

Week 8: Variable Selection

Tuesday, March 8, 2022 4:32 PM

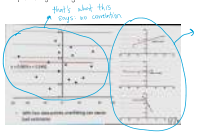
We've worked with factor-based models in classification, clustering, and regression.
We know how they work, how to build it, and how to interpret its results.
We also assume these models will have a lot of factors. but that's not always a good idea

Why limit the number of factors in the model?

Two reasons.

1) **Overfitting**, aka fits more closely to random effects than real effects
   → happens when # of factors ≥ # of data points

ex) Predict how tall a girl will be as an adult
Factor: how many degrees above or below the daily avg temp was 37 days after birth
→ I mean, that's just silly. There should be no correlation between the two factors



... but if data, then regression says there's a correlation.
Obviously this is wrong

Overfitting:
- ✓factors and ↓data
- ↑factors and ✓data
- ✓factors and ✓data = ↓factors and ↑data

↳ This is why we need to prevent overfitting.

2) **Simplicity**, usually better than complex models.
   → ↑complexity = ↑data collection = ↑difficulty & ↑costs
   ⇒ Less factors mean less chance of including insignificant factors
   i.e. p-value = 0.05 still means the factor is 5% chance. It's meaningless...
   If there are 20 factors, there's a 64% chance one is meaningless.
   ⇒ Easier to explain and interpret... after all, we want people to use the model, and they (managers/executives/others without an analytics background) won't want to use it without understanding it
   .....No matter how good or pure the data science is, if you can't clearly communicate what your model is doing, it just won't be trusted and thus not used.
   Most people don't want to know the R^2 value.
   They want to know what the R^2 value implies about the factors, which can be hard to do if there are hundreds of factors

We can choose factors by "variable selection"

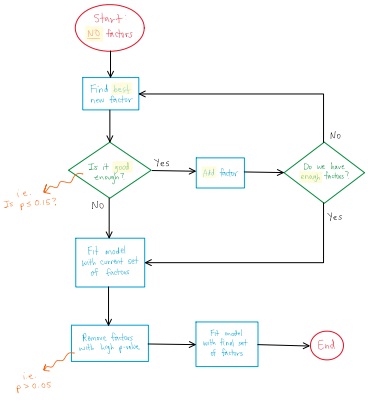
Greedy & Global approaches....
- Greedy: at each step, take one thing that looks best; future options are not considered.
- Global: all options looked at at the same time.

Greedy
(1) Forward Selection → start with no factors → continually add the best factors... until no more good factors to add
(2) Backward Elimination → start with all the factors → continually remove the worst factors... until no more bad factors to remove
(3) Stepwise Regression → combination of Forward & Backward

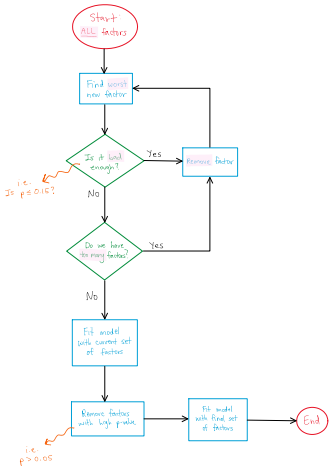
The "is it good enough?" can be p-value, R^2, AIC, etc.

Global
(4) LASSO
(5) Elastic Net
(6) ~~Ridge Regression~~ — not for variable selection

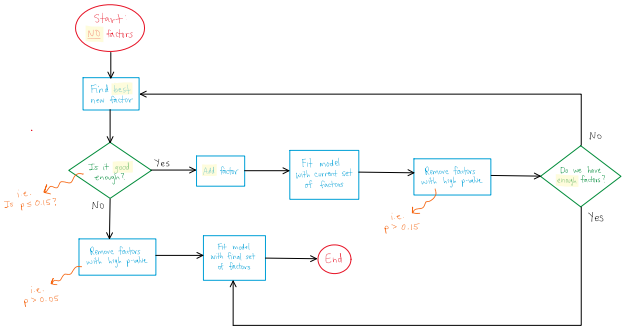
(1) Forward Selection



(2) Backward Elimination



(3) Stepwise Regression → forward approach example... lots of other approaches, though.



All of these are greedy...

Based on optimization modeling, there are other newer variable selection methods that look at all options at the same time.

↳ Global options

Recall regression ⇝ Minimize SSE to find the best Regression Line →

$$SSE = \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_{i1})]^2$$
$$= \sum_{i=1}^{n} [y_i - (a_0 + \sum_{j=1}^{m} a_j x_{ij})]^2$$

..... Remember to always scale the data first when constraining the sum of coefficients!
for (4), (5), (6)

(4) **LASSO**: add constraint to standard regression equation

(SSE) Minimize $\sum_{i=1}^{n} \left[y_i - (a_0 + a_1 x_{1i} + a_2 x_{2i} + \ldots + a_j x_{ji})\right]^2$

$\rightarrow$ Subject to $\sum_{i=1}^{j} |a_i| \le \tau$ $\rightarrow$ restriction: sum of coefficients can't be too large

$\hookrightarrow$ giving the regression a budget to use on coefficients .... use budget to prioritize important coefficients.... unimportant coefficients will be zero, so factor won't be used in the model.

How to pick τ? Depends on:

(1) Number of variables
(2) Quality of model as you allow more variables

} Use LASSO approach with many varying τ values to see which gives you the best trade-off

(5) **Elastic Net**: same as LASSO, but constrain a combination of the absolute values of the coefficients and their squares

Minimize $\sum_{i=1}^{n} \left[y_i - (a_0 + a_1 x_{1i} + a_2 x_{2i} + \ldots + a_j x_{ji})\right]^2$

$\Rightarrow$ Subject to $\lambda \sum_{i=1}^{j} |a_i| + (1-\lambda) \sum_{i=1}^{j} a_i^2 \le \tau$

NOT FOR VARIABLE SELECTION

(6) **Ridge Regression**: same as Elastic Net, but no absolute value term / $\lambda = 0$

Minimize $\sum_{i=1}^{n} \left[y_i - (a_0 + a_1 x_{1i} + a_2 x_{2i} + \ldots + a_j x_{ji})\right]^2$

$\Rightarrow$ Subject to $\sum_{i=1}^{j} a_i^2 \le \tau$

So, now we have 3 greedy approaches to variable selection, and 3 global ones.

Which method do we pick?

Pro/con comparison:

Greedy Methods
- Forward Selection
- Backwards Elimination
- Stepwise Regression ($\hookrightarrow$ more common among the three)

}
- Quick & fast
- Good for initial data analysis
- Doesn't check for random effects as well; not as thorough
- Thus, won't perform as well when tested on other datasets

Global Methods
- LASSO
- Elastic net

}
- More advanced
- Better predictive models
- Slower

Basically:
- If introductory data exploration: use greedy (step-wise) methods first, then build a more refined model later using LASSO/Elastic Net.
- If not introductory data: use LASSO or Elastic Net

What's the difference between LASSO & Elastic Net? ....

Regularized Regression

Lasso (Some coefficients forced to 0 to simplify model)
- Minimize $\sum_{i=1}^{n} (y_i - (a_0 + a_1 x_{1i} + a_2 x_{2i} + \ldots + a_j x_{ji}))^2$

  Subject to $\sum_{i=1}^{j} |a_i| \le \tau$

} (LASSO) Linear: Helps decrease the number of factors with non-zero coefficients } picks important predictors and discards the rest $\rightarrow$ has zero-value coefficients

Ridge (Coefficients shrink toward 0 to reduce variance in estimate)
- Minimize $\sum_{i=1}^{n} (y_i - (a_0 + a_1 x_{1i} + a_2 x_{2i} + \ldots + a_j x_{ji}))^2$

  Subject to $\sum_{i=1}^{j} a_i^2 \le \tau$

} (Ridge) Quadratic: Pushes coefficients towards zero, but not equal to zero } shrinks the coefficients of correlated predictors towards each other $\rightarrow$ has only non-zero value coefficients (no zeros) $\rightsquigarrow$ ↑ bias, ↓ variance ... but trading-off bias for variance (choosing variance over bias) = better predictive models .... so ridge method = ↑quality / ↓error!

Elastic net
- Minimize $\sum_{i=1}^{n} (y_i - (a_0 + a_1 x_{1i} + a_2 x_{2i} + \ldots + a_j x_{ji}))^2$

  Subject to $\lambda \sum_{i=1}^{j} |a_i| + (1-\lambda) \sum_{i=1}^{j} a_i^2 \le \tau$

$\Rightarrow$ Elastic Net = LASSO + Ridge

LASSO:
- variable selection benefits
- forcibly makes some coefficients zero, even if irrational:
  i.e. if had to choose between 2 highly correlated values, it'll pick one and discard the other

  i.e. 2 effective tests, but one is expensive and the other isn't. LASSO may innocently choose the expensive one unless you detect this and manually remove cost factor before rerunning the model

Ridge:
- predictive benefits
- All coefficients no matter how large are shrunk towards each other, so really good predictors may be underestimated.

Analytics is an art.

Models provide insight and direction, but human intuition is also beneficial.

$\rightarrow$ Try all the methods, and use your best judgement when comparing results

Week 9: Design of Experiments, Probability-based Models

Friday, April 8, 2022 11:33 PM

So far this course, the data is given or is easily collected.

But that's not always the case.

Sometimes, we don't have data, or a full data set is hard to get.

i.e. Data can come from a survey, but surveying everyone in the world is expensive and impossible.
So, instead ask, like, 600 people, and from a representative sample.
↳ This is easier said than done.
Many dimensions (gender, age, race, etc), and not so many data points.

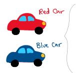
Design of Experiments (DOE):

designing a way to collect the best subset of data that answers our questions, quickly and efficiently.

Concepts:

(1) Comparison
(2) Control
(3) Blocking

ex)



(1) Comparison: compare red car [factor] to blue car [factor]
i.e. comparing red car price to blue car price

(2) Control: making sure that all other factors outside the comparison are the same
i.e. Good control: comparing 1 yr old red smart car to 1 yr old blue smart car
Bad control: comparing 1 yr old red sports car to 10 yr old blue smart car

(3) Blocking: a blocking factor is something that could create variation
i.e. type of car is a blocking factor that could account for some of the differences between red/blue cars because sports cars are more likely to be red. Instead, analyze red sports cars by themself. It'll lower variance instead of analyzing red cars altogether.

A few different types of DOE....

A/B Testing: a test to see which of two alternatives is better

When choosing between 2 alternatives, can use A/B testing as long as these are also true:

1. Collect data quickly enough to get an answer in time.
2. Data must come from a representative sample of the total population.
3. Amount of data is small compared to the total population.
   i.e. If expecting only 2500 views for ad, and we spend the first 2000 views collecting data, then there won't be enough people left for us to get much benefit from the results.

ex) Which of the two ads are more effective?

A)
B)

Suppose we use both ads for the first 2000 times the ad is shown. Each time, it randomly shows ad A or ad B, and we track how often it gets clicked.

Note that we could do hypothesis testing on the fly; any time significant difference is detected, stop testing and use better option.

Scenario 1

→ Hypothesis test: B is significantly better

Scenario 2

→ Hypothesis test: No significant difference.

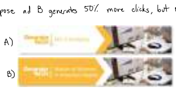
But what if we have several alternatives, not just two?
& If different factors/differences involved, we want to know why A is better than B? ....

Factorial Testing: a test to see which of many alternatives is better

Break the comparisons into factors rather than an absolute A and B.
.... If you're in the position to decide what data to collect and what comparisons to try, factorial design can be a powerful tool.

ex) Suppose ad B generates 50% more clicks, but why?

A)
B)



Different fonts, different wording, different background color....
These are all factors we can test.

Full factorial Design: test the effectiveness of every combination
→ tests all combinations

Fractional Factorial Design: test the effectiveness of a subset of combinations
→ estimates all effects by comparing some combinations

Full Factorial Design ⇒ 8 combinations (ALL)

Font	Wording	Background
Arial	MS in Analytics	White
Arial	MS in Analytics	Gold
Arial	Master of Science in Analytics	White
Arial	Master of Science in Analytics	Gold
Roboto	MS in Analytics	White
Roboto	MS in Analytics	Gold
Roboto	Master of Science in Analytics	White
Roboto	Master of Science in Analytics	Gold

Fractional Factorial Design ⇒ 4 combinations (subset)

Font	Wording	Background
Arial	MS in Analytics	White
Arial	Master of Science in Analytics	Gold
Roboto	MS in Analytics	Gold
Roboto	Master of Science in Analytics	White

→ Fractional factorial design: test subset of combinations

Balance Design
- Test each choice the same # of times
- Test each pair the same # of times

→ Full factorial design: test every combination of the ad
- 2 fonts × 2 wordings × 2 backgrounds = 8 combinations
  ↳ use ANOVA to determine importance of each factor

→ But what if there are many combinations?
- 7 factors, 3 choices each = 3^7 = 2187 combinations

.... That's WAY too many! → use fractional factorial design instead

If factors are independent (AKA one factor doesn't depend on another factor): test subset of combinations, then use regression to estimate the effect of each choice; each factor would get its own categorical variable.

ex) Variables { Background Color: gold, blue, white; Font Size: 10, 12, 14 } ⇒ Based on each of the factors that make up every combination, we could use regression to estimate the number of clicks as the response. This could work well if there aren't significant interactions between the factors.

May need to include Interaction Terms.
For example, font color and background color are probably not independent; a white background may do well on tests, and a white font may do well on tests but a white font against a white background would not be very effective.

Still, there is a trade-off:

More Information vs. Immediate Vale
↳ Exploration vs. Exploitation

This trade-off problem is called the Multi-Armed Bandit.
... name comes from gambling slot machines. To play the highest payout machine, the player would need to test all of them.

Iteration Process:

- Start with no information
  ↳ Equal probability of selecting each alternative
- Perform some tests
  ↳ Gather information
- Based on new information, choose next one on which one you think/know is the best
  ↳ Update Information

Repeat ...

Parameters in the iteration:
- Number of tests between recalculating possibilities
- How to update the probabilities

Collect data.
Update possibilities.
Continue until you've found the best one.

Very good for learning faster on the fly and creating more value along the way.
Worthwhile! Better than running a fixed, large number of tests.

Probability Distributions → software can determine if output fits probability distribution

Sometimes you don't need the complexity of factors in a model.
Sometimes a simple model is all you need.
↳ Probability distributions can form the backbone of such simple models → it's the simple approach..... i.e. baseball example in lecture
Especially true when the only data we have is the response, and/or it would be hard to collect/analyze additional information.

→ **Bernoulli Distribution:** probability of yes/no (success/fail) with the outcome of the single trial of the event → i.e. coin flip

→ consists of one trial

Probability Mass Function:
$$P(X = 1) = p$$
$$P(X = 0) = 1 - p$$

independent

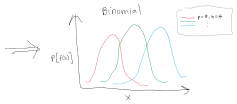
→ **Binomial Distribution**: probability of yes/no (success/fail) with the outcome of the multiple trials of the event ⟶ i.e. count of successful coin flips in a trial

⟶ consists of many Bernoulli(p) trials ⟶ probability of x successes out of n Bernoulli(p) trials

Probability Mass Function:

$$P(X=x) = \binom{n}{x} p^x (1-p)^{n-x}$$
$$= \left(\frac{n!}{(n-x)!\,x!}\right) p^x (1-p)^{n-x}$$

⟹ As n, binomial distributions converges to normal distribution



→ **Geometric Distribution**: probability of # of failures before the first success ⟶ i.e. unsuccessful coin flips before first success OR fail-to-success ratio of door-to-door sales

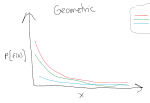
⟶ consists of many Bernoulli(p) trials ⟶ probability of having x Bernoulli(p) failures until the first success
OR probability of having x Bernoulli(1-p) successes until the first failure

Probability Mass Function:

failures until first success: $P(X = x) = (1-p)^x p$

successes until first failure: $P(X = x) = (1-p) p^x$

⟹ Can also use geometric distribution to check if factors are independent and identically distributed (i.i.d.)

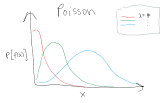


independent

→ **Poisson Distribution**: probability of having a certain # of events occurring in a fixed time interval ⟶ i.e. # of customers arriving in a restaurant

Probability Mass Function: $f_x(x) = \frac{\lambda^x e^{-\lambda}}{x!}$

- Good at modeling random arrivals
- λ: average number of arrivals per time period
- Arrivals are independent and identically distributed (i.i.d.)



If Poisson then it's **Memoryless**: distribution of time to next interval = initial distribution of time to next arrival
(& vice-versa)
↳ memoryless: it doesn't matter what happened in the past, all that matters is where we are now
↳ if not memoryless, then NOT Poisson

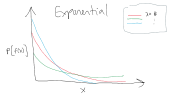
independent

→ **Exponential Distribution**: probability of time waiting before an event occurs ⟶ i.e. time between customers arriving in a restaurant on a given day

Probability Mass Function: $f_x(x) = \lambda e^{-\lambda x}$

If arrivals are Poisson(λ), then the time between successive arrivals is exponential(λ) distribution.

⟹



If exponential then it's **Memoryless**: distribution of remaining time = initial distribution of time
(& vice-versa)
↳ if not memoryless, then NOT exponential

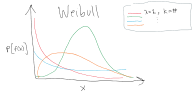
→ **Weibull Distribution**: probability of the time in which something fails ⟶ i.e. lifecycle curve of phone battery before recommended to be replaced

⟶ Weibull vs. Geometric
(time between failures) vs. (# of tries between failures)

Probability Mass Function: $f_x(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k}$

λ: Scale Parameter
k: shape parameter

⟹



- $k > 1$ ⇒ modeling when failure rate decreases with time; "worst things fail first" i.e. parts with defects
- $k > 1$ ⇒ modeling when failure rate increases with time; "things that wear out" i.e. tires
- $k = 1$ ⇒ modeling when failure rate is constant with time...
  If $k = 1$, then it becomes an exponential distribution.

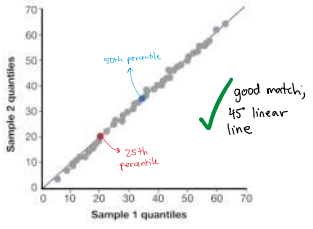
$$f_x(x) = \left(\frac{1}{\lambda}\right)\left(\frac{x}{\lambda}\right)^{0} e^{-\left(\frac{x}{\lambda}\right)^1}$$
$$= \frac{1}{\lambda} e^{-\frac{x}{\lambda}} = \lambda e^{-\lambda x}$$
(replace 1 with k)

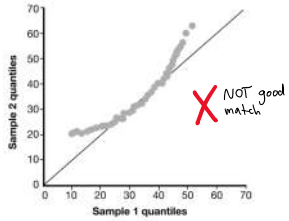
Q-Q Plots: visualization on whether two distributions of data are about the same, or whether one data set is distributed similarly to a probability distribution.

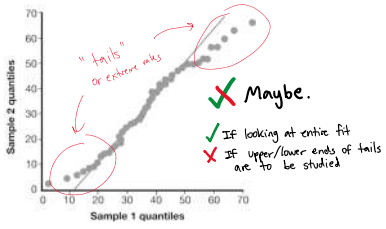
Q-Q plots at percentile/quantile of data sets to show that they are/are not distributed the same.
The idea is that, despite any variations, two similar distributions should have about the same value at each quantile.

Plotting (x, y), where:

x = Sample 1's zth percentile value
y = Sample 2's zth percentile value







Alternatively, x-axis is data
& y-axis is ideal/theoretical values for perfect distribution.

Queuing Models: study of waiting lines / queues.

Kendall Notation ("A/M/1 queue")

Potential queuing model parameters

- General arrival distribution [A]
- General service distribution [S]
- Number of servers [c]
- Size of the queue [K]
- Population size [N]
- Queuing discipline [D]

example for a call center

- Arrival Rate (calls) = λ
- Service Rate (calls) = $\mu > \lambda$
- Transition Equations (≥ 1 calls in the queue)
  - P(Next event is an arrival) = $\frac{\lambda}{\lambda + \mu}$
  - P(Next event is finished call) = $\frac{\mu}{\lambda + \mu}$
- Can calculate:
  - Expected fraction of time employee is busy = $\frac{\lambda}{\mu}$
  - Expected waiting time before talking to employee = $\frac{\lambda}{\mu(\mu - \lambda)}$
  - Expected number of calls waiting in queue = $\frac{\lambda^2}{\mu(\mu - \lambda)}$

Simulation: build a model to watch its behavior

Pick 1

- **Deterministic Simulations**: no randomness: never vary; same inputs results in same outputs
- **Stochastic Simulations**: has randomness: varies; different inputs results in different outputs

Pick 1

- **Continuous-Time Simulations**: changes happen continually (often modeled with differential equations)
  i.e. chemical processes, propagation of diseases
- **Discrete Event Simulations**: changes happen at discrete time points
  i.e. call center simulations, airport departure, car wash

Focus on lesson: **discrete-event stochastic simulations**

- Valuable when systems have high variability/randomness
- Using average values isn't good enough
  - ↳ Use a lot of replications (number of runs)
    1 replication = 1 data point (may be unrepresentative)

Simulation Validation

- Use real data to validate your simulation is giving reasonable results
  - Real and simulated averages don't match → problem
  - Averages match, variances don't match → problem

Simulation

- Simulation software
  - Elements of model include
    - Entities: things that move through simulation (e.g., bags, people, etc.)
    - Modules: parts of process (e.g., queues, storage, etc.)
    - Actions
    - Resources (e.g., workers)
    - Decision points
    - Statistical tracking
    - Etc.
  - Often "drag and drop" programming style
  - Under the hood
    - (Pseudo) random number generation, tracking, animation, etc.

We have learned about descriptive & predictive analytics.
When talking about simulation, we can now discuss prescription analytics, also the "what if?" & "what should I do?" questions

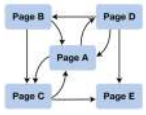
Samples

- how are the overall process throughput change if the company invested a $100,000 on a faster machine for one step of the process?
- how valuable would hiring an extra call center worker be?
- what's the right distribution of baggage tugs to have at the airport – one for each gate, one for every two gates, or a floating pool at each half side of the concourse?

Simulation can be a powerful prescriptive tool.
But remember, it's only as good as its input & assumptions.

Markov Chains: stochastic model describing a sequence of possible memoryless events

- Memoryless: state transitions only depend on the most recent state
- Most systems don't exhibit this property, but if they do, it's a powerful tool



ex)

Can simulate into a steady-state to figure out what % probability it would be on the pgs

Week 10: Missing Data, Optimization

Saturday, April 9, 2022 10:28 PM

So far, we assume we have data. Sometimes it's big data — billions of data points.
But sometimes, not all of those data points are available.
Sometimes there is missing data.

Example →

Car #	Color	Style	Miles	Price
1	White	Sedan	15,000	$15K
2	White	missing	30,000	$12K
3	Black	Sedan	37,000	$10K
4	Gray	Coupe	18,000	$12K
5	Red	SUV	missing	$20K
6	White	Pickup	24,000	$15K
7	Gray	Sedan	23,000	$17K

Can just be missing or wrong, but also there could be a pattern.

i.e. Missing data may occur during:
- Temperature Readings → gauges break down
- Filling out forms → incorrectly filled out, purposely left blank, unavailable data
- Income Surveys → higher income people less likely to report income ← BIAS
- Radar Gun → may treat slow cars as anomaly and not show a speed

What to do with missing data?

(1) Throw away
(2) Use categorical variables ("N/A", "missing") to indicate missing data
(3) Impute/estimate data

(1) Throw Away

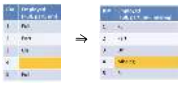
Pros
- Not potentially introducing errors
- Easy to implement

Cons
- Don't want to lose too many data points
- Potential for censored or biased missing data → throwing out potentially biased datasets = more bias

(2) Categorical Variable Approach

→ Categorical variable has missing data:

Add category: missing



→ Quantitative variable has missing data:

- All missing values = 0
- Add new categorical variable

(3) Imputation

No more than 5% of data should be imputed ..... to avoid overfitting (data used twice)

Imputation of more than one factor value for a data point is possible:
Multivariate Imputation by Chained Equations (MICE)

3 methods:

1. Midrange Value

Mean, Median (numeric), Mode (categorial)

Advantages:
- Hedge against being too wrong
- Easy to compute

Disadvantages:
- Imputation can be biased
  i.e. means/mode will be underestimated in income survey since people with high income is less likely to answer

2. Use a Predictive Model

Predicting the value of missing data based on the other variables
...like regression.

Advantages:
- Reduces / Eliminates bias

Disadvantages:
- Complex: build fit, validate, & test to estimate missing value
- Overfitting: using the data twice; once for imputation, second to fit model
- Imputation does not capture all variability

3. Perturbation

+/- random amount from a normally-distributed distribution
..."noise" or random value

Advantages:
- More accurate variability
  (Artificially brings in variability)

Disadvantages:
- Less accurate on average (better off using predictive models)

Question: Does imputation introduce additional errors?

- Imputation Error + Perturbation Error (if applicable) + Model Error
  - It's very hard to determine each error component, and how they add up.
  - We can use a test data set to evaluate the overall model's performance.
  - That's the most ideal solution, despite the fact that the test data is missing some data
- So, yes, errors do get added.
- That doesn't mean imputation is inferior to using regular data.
  - Regular data can be just as worrisome
    - It's almost impossible to have a dataset where each value is actually correct.

Data is always imperfect:

- Errors
- Outliers
- Missing data

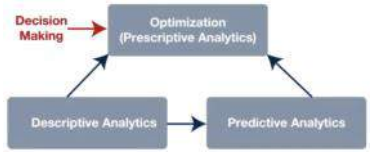
Optimization

Optimization is used by many statistical models (clustering, regression, variable selection, time series...) to get answers in descriptive and predictive analytics.

But optimization can also be used for prescription analytics: "given what I know and what I predict, what's the best course of action to take?"

Optimization can answer questions like:

- Which airplane mechanic should be scheduled for each shift over the next week to meet the expected maintenance requirements at O'Hare Airport at lowest cost, making sure we don't violate any federal regulations or union contract requirements and accounting for unexpected breakdowns?
- How much crude oil should be sent by tanker and by pipeline from each oil field to each refinery to meet demand and avoid oversupply while keeping costs low?
- What web pages should be cached in a server farm, and how many copies of each should be stored to maximize the profit made by responding quickly to hits?
- How should a large machine shop sequence its production to get maximum throughput while meeting all clients' deadlines, taking into account the reality that some output will fail inspection and need to be remade?
- What's the shortest route from my house to the airport, given current and predicted traffic?



Predictive/Descriptive Models:
Software automates solution
Software automates model building

Optimization/Prescriptive Models:
Software automates solution
YOU do the model building → building models requires human insight & expertise

Optimization models have three main components:

1) Variables

Decisions to be made.

Example: political candidate scheduling
x_i = total time spent in state i
y_i = number of visits to state i
z_i = 1 if state i is ever visited, 0 if not
w_{id} = time spent in state i on day d
v_{id} = 1 if state i visited on day d, 0 if not

2) Constraints

Restrictions on variable values.

Constraints:
30 days left in campaign
$\sum_i x_i \leq 30$
At least 3 Florida visits in days 24-30
$\sum_{d=24}^{30} v_{Florida,d} \geq 3$
Total visits must add up correctly
$\sum_d v_{id} = y_i$

3) Objective Function

Solution quality measure; what are you trying to minimize/maximize?
Can rank/assess variables given restrictions.

$$\sum_i \left(\alpha p_i \sqrt{x_i + \frac{1}{3}\sum_{j \in N(i)} x_j} + \beta v_{id} f_d \right)$$

Determined by other analytics models!

Pretty common for optimization models to require input that's found as the output from other models

Some more definitions...

Solution: values for each variable
Feasible Solution: variable values that satisfy all constraints
Optimal Solution: feasible solution with the best objective value

Modeling is an Art (not a science): Examples

Diet Problem (US Army):

Diet Problem (US Army):

Satisfy soldiers' nutritional requirements at minimum cost

Given:

- n foods
- m nutrients
- a_{ij} = amount of nutrient j per unit of food i
- m_j = minimum daily intake of nutrient j
- M_j = maximum daily intake of nutrient j
- c_i = per-unit cost of food i

→ Variables: (Decisions)

- x_i = amount of food i in daily diet

→ Constraints:

- $\sum_i a_{ij} x_i \geq m_j$ for each nutrient j
- $\sum_i a_{ij} x_i \leq M_j$ for each nutrient j
- $x_i \geq 0$ for each food i

→ Objective Function:

- Minimize $\sum_i c_i x_i$

✓ Optimization Model.

Other complexities to consider:

- Variety in diets
- Seasonal cost variation
- Taste of food
- Taste of combinations of foods

Example 2

– Call center scheduling

- Meet forecasted demand d_i for each day of the week i
- Workers work 5 days in a row, then 2 days off
- Minimize worker-days used

Variables
x_i = number of people working on day i
Objective function
minimize $x_{Sunday} + x_{Monday} + \dots + x_{Saturday}$
Constraints
Meet demand: $x_i \geq d_i$ for all days i
Integer workers: x_i is integer for all days i
5-day requirement: ???

Variables
x_i = number of people who *start* working on day i
Objective function
minimize $5(x_{Sunday} + x_{Monday} + \dots + x_{Saturday})$
Constraints
Meet demand: $\sum_{j \text{ working on day } i} x_j \geq d_i$
Ex: $x_{Fri} + x_{Sat} + x_{Sun} + x_{Mon} + x_{Tue} \geq d_{Tue}$
Non-negativity: $x_i \geq 0$ for all days i
Integerality: x_i is integer for all days i

Modeling with Binary Variables:

Example: Stock Market Investment

Invest to balance return and risk

B = investment budget
n = number of stocks available
r_i = expected return of stock i relative to market
Q_{ij} = covariance of returns of stocks i and j
m_i = minimum dollar amount for each stock

Personal constraints (more examples)
If invest in energy, invest in at least 5

Option 2: z_{Energy} = 1 if invest in energy, 0 if not

$$\sum_{j \in \text{energy}} y_j \geq 5 z_{Energy}$$

$z_{Energy} \geq y_i$ for all energy stocks i

Variables
x_i = amount invested in stock i
y_i = 1 if invest in stock i, 0 if not
Constraints
$\sum_i x_i \leq B$
$x_i \geq 0$ for all stocks i
$x_i \leq B y_i$ for all stocks i
$x_i \geq m_i y_i$ for all stocks i
$y_{Tesla} = 1$
$y_{Amazon} + y_{Google} + y_{Apple} \geq 1$
$y_{FedEx} = y_{UPS}$
$y_{Coca\text{-}Cola} = 1 - y_{PepsiCo}$
Objective function
Maximize $\sum_i r_i x_i - \theta \sum_i \sum_j Q_{ij} x_i x_j$
$- \sum_i t y_i$

Sunday, April 10, 2022 12:08 PM

Optimization is important in analytics/statistics.

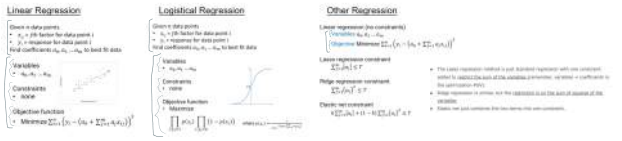
In fact, it's used in all of our modeling techniques thus far.

Here are examples.

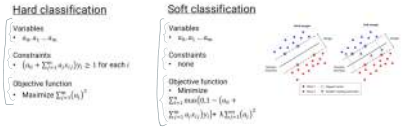
First, a note...

Notation confusion: statistics vs. optimization

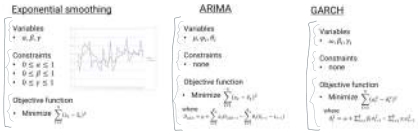
Regression



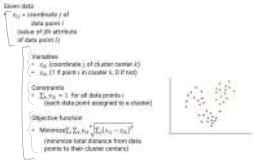
Support vector machine models



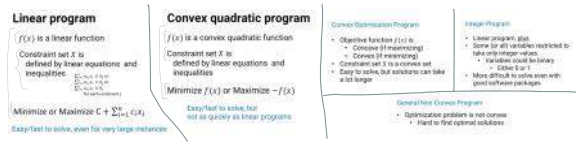
Time series models



k-means for clustering



Classification of Optimization Models



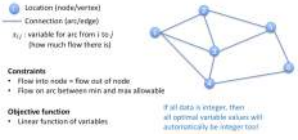
Objective Function	Constraint	Program Type	Difficulty to Solve	Time to Solve
Linear	Linear	Linear	Easy	Fast
Quadratic	Linear	Convex Quadratic	Easy	Fast, but slower than Linear Objective Function
Quadratic	Quadratic	Convex	Easy	Can Take Long
Linear/Quadratic	Linear, but some variables are Binary	Integer Program (subset of Linear)	Hard	Slow
Linear/Quadratic	Linear, but some variables are Integers	Integer Program (subset of Linear)	Harder	Slower
		General Non Convex	Hardest	Slowest

From quickest/easiest to slowest/hardest:

1. Linear Programs
2. Convex Quadratic Programs
3. Convex Programs
4. Integer Programs
5. General Non-Convex Programs

If program is too hard, use a heuristic.

Network Models (type of linear program)



Examples:

- Google Maps: find quickest/shortest route from one place to another
- Assignment Model: which worker gets which job to maximize workforce efficiency?
- Maximum Flow Model: how much oil can flow through complex network of pipes?

Stochastic Modeling

A model that accounts for randomness or uncertainty.

Use if:
- data/parameter isn't exactly known
- forecast value isn't exactly known

Two Approaches:

1. Model Conservatively

1. Model Conservatively

2. Scenario Modeling

Programming Models

The models we have seen so far are **mathematical programming models**.
As in, they have variables, constraints, and an objective function.

But other models may have a different structure:

Dynamic Program

- Assumes there is no uncertainty
- Relies on states (the exact situation, and their values) and decisions (choices of next state).
- Systems are divided into states. The state specifies exactly what's going on in the system.
- Bellman's Equation: determine optimal decisions

Stochastic Dynamic Program

- Introduces uncertainty
- Basically dynamic programming model, but decisions have probabilities of next state

Markov Decision Process

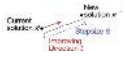
- Stochastic dynamic program with discrete states and decisions
- Probabilities depend only on current state/decision.

Solving Optimization Models

We've learned how to build optimization models. Now we'll learn how to solve them.

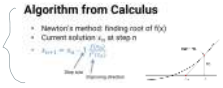
Most optimization algorithms have these main steps:

1. **Initialization**
   Creating a first/starting solution, no matter how simple/bad it is.
2. **Iterate**
   Repeat:
   - Find an improving direction t
   - Using a step size Θ to move along it
   - New solution = old solution + Θt

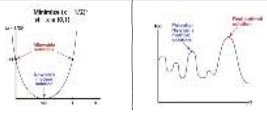


3. **Stop**
   ...when solution doesn't change much, or time runs out.

Example



Convex Optimization Problem	Non-Convex Optimization Problem
Guaranteed to find optimal solution	NOT Guaranteed to find optimal solution
	May Converge to infeasible Solution
	May find only a local optimum



Non-Parametric Models

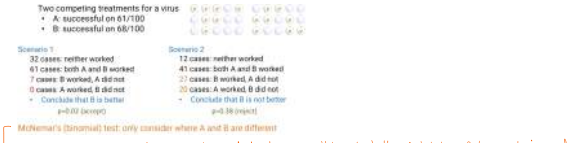
Typical hypothesis testing assume that we know the underlying distribution we're testing against.
But that's not always the case. Sometimes all we have is a response and we don't have a good model for the distribution.

Non-parametric tests: use when nothing is known about underlying distribution.
Distribution-free data, but we need just enough data to figure out where the middle value is

McNemar's Test

Used for comparing results on pairs of responses/data points where 2 approaches were used on the same thing.

Does the following:
1. Throw out all the cases where the results are the same
2. Tests (non-distributed) responses using the binomial distribution to see whether we'd expect results this extreme, or more extreme by luck.



SUMMARY

Non-Parametric Test	Assumptions	How It's Done Other Notable Attributes of Test Uses	Other Notable Attributes of Test Uses	Uses

McNemar's (binomial) test: only consider where A and B are different

↳ To use McNemar's test, we don't need to know anything about the distribution of how much is cured. It's just a comparison of a pair of results.

Wilcoxon Signed Rank Test for Medians

Answers the following questions:
- Is the median of the distribution different from m?
- Do two sets of paired samples have the same median?

Does the following:
1. Rank $|y_1 - m|, \ldots |y_n - m|$ from smallest to largest
2. $W = \sum_{y_i > m} rank(y_i - m)$ = sum of all ranks where $y_i > m$
3. Get p-value test for W

↳ Answering whether ranks indicate that the two sets of observations are likely to be different

Mann-Whitney Test

Used for two data set analysis, but not paired samples.

Does the following:
1. Rank all observations together
2. [illegible]
3. Find significance of U (need software or table)

Summary:

Use parametric tests when nothing is known about its underlying distribution.

Two Data Sets:
- McNemar's Test (paired yes/no)
- Wilcoxon Signed Rank Test (paired numeric data)
- Mann-Whitney (unpaired)

One Data Set:
- Wilcoxon Signed Rank Test (compare possible median)

SUMMARY

Non-Parametric Test	Assumptions	How It's Done Other Notable Attributes of Test Uses	Other Notable Attributes of Test Uses	Uses
McNemar's Test	None needed. This method doesn't care about the underlying distribution	1. Throws out all the cases where the results are the same. 2. Tests the responses (NOT the underlying distribution) using the binomial distribution to see whether we'd expect results this extreme or more extreme just by luck	We don't need to know anything about the distribution. It's just a comparison of the results	Used for comparing results on pairs of responses, data points where two different approaches were used on the same thing.
Wilcoxon Signed Rank Test for Medians	Distribution is continuous and symmetric	**Median difference Test** Given responses $y_1, \ldots y_n$: 1. Rank $\lvert y_1 - m\rvert, \ldots \lvert y_n - m\rvert$ from smallest to largest 2. $W = \sum_{y_i > m} rank(y_i - m)$ only for y_i values where $y_i > m$ 3. Get p-value test for W **Sample Median Equivalence Test (paired test)** Given pairs $(y_1, z_1), \ldots (y_n, z_n)$ from observations y & z, use $\lvert y_1 - z_1\rvert, \ldots \lvert y_n - z_n\rvert$ for rank test	Calculating p-values here are harder than McNemar's Test. The Wilcoxon test is like a normal distribution test	Answers the questions: 1. Is the median of the distribution different from m? 2. Do two sets of paired samples have the same median?
Mann-Whitney Test	Samples are Independent from each other	Given independent observations $y_1, \ldots y_n$ and $z_1, \ldots z_m$: 1. Rank all observations **together:** $y_1, \ldots y_n, z_1, \ldots z_m$ 2. U = smaller of two adjusted rank sums: $U = min(U_y, U_z)$ where: $U_y = \sum_{i=1}^{n} rank(y_i) - \frac{n(n+1)}{2}$ $U_z = \sum_{j=1}^{m} rank(z_j) - \frac{m(m+1)}{2}$ 3. Find significance of U (need software or table)	--	Used for Two Data set analysis, but not paired samples

Bayesian Models

Based on conditional probability, aka Baye's Theorem $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

"The probability of A given the probability of B given A, times the probability of A, divided by the probability of B."

The problem with Bayes' Theorem is that it can feel counter-intuitive.

Example: Medical test for a disease
- True positives: 98%
- False positives: 8%
- 1% of population really has disease
  - 8.9% of people test positive
- If someone tests positive, what is the probability they have the disease?

- A: has the disease
- B: tested positive
- $P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{98\% \times 1\%}{8.9\%} = 11\%$
- After testing positive
  - a person has only an 11% chance of having the disease

Why?
So many more people don't have the disease...
...so there are many more false positives than true positives.

Empirical Bayes Modeling

Useful when an overall distribution of something is known or estimated, but there's only a little data available for a specific case.

Example:
- predicting outcomes of the NCAA basketball tournament
- Difference X in points scored by the home team and road team
  - approximately normal: $X \sim N(m + h, \sigma^2)$
  - h: home court advantage
  - m: true difference in the teams' strength (unknown)
  - σ^2: variance
  - The differences between college basketball teams' strength
    - $m \sim N(0, \tau^2)$

→

- Observed data
  - x = observed point difference in game
  - m = real difference between the two teams, $m \neq x$
- $P(M = m|X = x) = \frac{P(X=x|M=m)P(M=m)}{P(X=x)}$
- $M|X = x \sim N\left(\frac{\tau^2}{\tau^2+\sigma^2}(x-h), \frac{\tau^2\sigma^2}{\tau^2+\sigma^2}\right)$
- Get σ, τ, h from the past data
- $P(\text{home team better} \mid X = x) = \int_0^\infty P(M = m|X = x)\,dm$

→

- Example: home team won by 20 points
  - Estimate home-court advantage h = 4 points
  - Standard deviation in team strength difference τ = 6 points
  - Standard error from random variation σ = 11 points
- Of the 20 point victory
  - About 4 points due to home court
  - About 12.5 points due to random variation
  - Only 3.5 points due to the difference between teams
- A lot more variance due to randomness

In summary:

We can: 1) Take a single observation,
2) Combine with a broader set of observations that are relevant in general (but not specifically),
3) Then make a deduction/prediction.

So, Bayesian models work especially in the absence of lots of data.

Communities

An analysis of large, interconnected populations may have impacts on:
- marketing messages propagate through social media networks
- disease outbreaks spread through populations
- computer viruses spread through a network
- words have spread from language to language over time
- terrorist networks communicate
- and more...

A look into highly-connected subpopulations...

Here's a graphic where each person is a circle, and a line between two circles means those two people are in the same room together at least once a day

If the red person gets the flu, it may take a long time to spread all the way to the orange person; the shortest way requires four connections.

On the other hand, look at all the pink people. These people are much more likely to get the flu quickly and spread it.

The same is true if we were talking about getting a marketing message through a social network.

So, this can be a powerful modeling tool.

Some Definitions....

Community: a set of circles that's highly connected within itself

Graph
- Circles = nodes/vertices
- Lines = arcs/edges
- Clique (pick) = a set of nodes that all have edges between each other

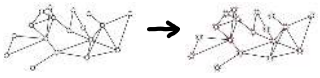
Louvain Algorithm: heuristic to decomposing a graph into communities

Maximize the modularity of a graph

- a_{ij}: weight on the arc between nodes i and j
- w_i: total weight of arcs connected to i
- W: total weight of all the arcs
- Modularity = $\frac{1}{2W}\sum_{i,j \text{ in same community}} \left(a_{ij} - \frac{w_i w_j}{2W}\right)$

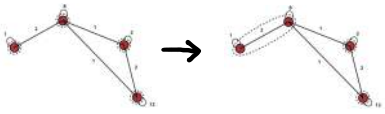
Steps:

Step 0
Each node is its own community



Step 1
Repeat...
Make biggest modularity increase by moving a node from its community to an adjacent node's community
...until no move increases modularity

Step 2
Each community is a super-node
Repeat Step 1 using super-nodes



Basically, whichever move creates the biggest modularity increase, that's where node i goes. Or if nothing increases modularity, node i just stays where it is. It does that for every node and then starts again with the first node and goes through them all again until there's no way to improve modularity by moving a node.

Neural Networks & Deep Learning

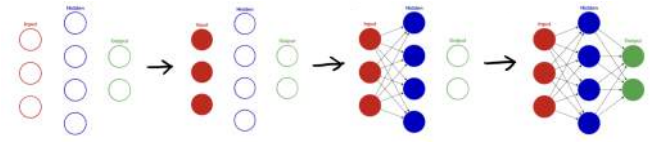
Training system to react to patterns human brains can react to, even if we don't understand the pattern ourselves.

Used in:
- Natural Language Processing
- Speech recognition
- Image Recognition

3 levels of neurons:
1) Input Level
2) Hidden Level
3) Output Level

Each neuron:
- Gets inputs from previous layer
- Calculates function of weighted inputs
- Gives its output to next layer

Deep Learning = many deep layers



Competitive Models

Descriptive models: get an understanding of reality
Predictive models: find hidden relationships; predict the future
Prescriptive models: find the best thing to do

Assumes the system does not react

.... But what if the system reacts, and it reacts intelligently?

Examples

→ Use analytics to consider all sides to the system.

Game Theory: competitive decision-making

Cooperative Game Theory: competition + cooperation

Need to think of varying strategies:

Strategy (use optimization model to figure out optimal strategy)

- counter-strategies for competitors
- counter-counter strategy (etc.)
- Sequential game: see other side's strategy before making your decision (vice versa)
- Pure Strategy: just do one thing
- Mixed Strategy: randomized strategy (according to probabilities)
  - ↳ i.e. rock-paper-scissors

Information

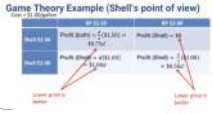
- Perfect information: know all about everyone else's situation; balanced/symmetric, i.e. chess
- Imperfect information: Some have more information than others; unbalanced/asymmetric, i.e. competing restaurants

Sum

- Zero-sum: whatever one side gets, the other side loses, i.e. +1 or -1 loss like rock-paper-scissors
- Non-zero-sum games: total benefit might be higher or lower, i.e. economy

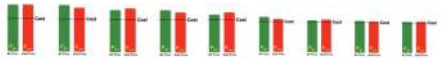
Game Theory Example

- Two gas stations
- BP VS Shell
  - Set price: $2.50 or $2.00
    - Same price: 50/50 demand
    - Otherwise: all demand to lower-priced one
  - What's the best price to choose?



Stable Equilibrium

- Neither station has incentive to change
  - In this case...
    - Outcome is worse than if they had agreed to both charge the higher price
    - "Prisoner's dilemma"
      - Even if agree on higher price, both sides have incentive to cheat
      - So both end up with lower price
    - Not always true in game theory



Extension of Example

- Choose any price they want
- BP price = P_{BP}
  - Shell if $p_{Shell} > P_{BP}$ then profit = SP (will undercut)
    - If $P_{Shell} = P_{BP}$ then profit = $\frac{1}{2}(P_{Shell} - cost)$
    - If $P_{Shell} < P_{BP}$ then profit = $d(P_{Shell} - cost)$
  - So, Shell might price slightly lower price than BP
    - Then, BP might price slightly lower than Shell
      - Then Shell might price slightly lower than BP
      - Etc.
- Both keep lowering prices until price is about equal to the cost

Key Notes

Saturday, April 30, 2022 2:35 PM

Maybe small sample size...
↑factors, ↓data → use ridge regression

Maybe linear regression fits data too well.
Ridge regression finds a new line that doesn't fit the training data as well.

(Least Squares)
Linear Regression: no bias ⇒ but, ↑ variance

Ridge Regression: ↑ bias ⇒ so, ↓ variance

Lasso Regression can exclude useless variables → good when so many factors
Ridge Regression more useful if we know all the factors are important.
Elastic-Net Regression good if we don't know the situation since EL = Ridge + Lasso (best of both worlds)

Classification tree → true/false (discrete) leaves
Regression tree → numeric data leaves

Logistic Regression → discrete values (yes/no) for axis (y-axis namely)

SVM: maximize gap (margin)